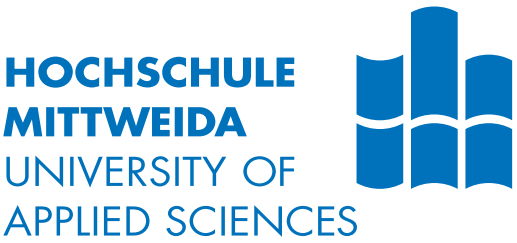


Loeper, Marcus

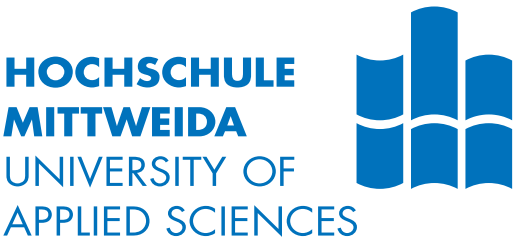
Analyse und Entwicklung von 2D-Charakteranimations-werkzeugen in Adobe After Effects

– Bachelorarbeit –

Hochschule Mittweida – University of Applied Science

Fachbereich Medien

Berlin – 2011



Loeper, Marcus

Analyse und Entwicklung von 2D-Charakteranimations-werkzeugen in Adobe After Effects

– eingereicht als Bachelorarbeit –

Hochschule Mittweida – University of Applied Science

Fachbereich Medien

Erstprüfer	Zweitprüfer
Prof. Dr.-Ing. Robert J. Wierzbicki	Dipl.-Ing. Sieglinde Klimant

Berlin – 2011

Bibliographische Beschreibung

Loeper, Marcus:
Analyse und Entwicklung von 2D-Charakteranimationswerkzeugen
in Adobe After Effects – 2011 – 74 S.
Berlin, Hochschule Mittweida, Fachbereich Medien, Bachelorarbeit

Referat

Diese Bachelorarbeit untersucht die Notwendigkeit eines 2D-Charakteranimationstools für Adobe After Effects. Welche Möglichkeiten bietet After Effects im Moment dem Nutzer an, um Charaktere bequem zu animieren? Daraus schlussfolgernd sollen die essentiellen Features und deren Entwicklung unter Berücksichtigung der Bedienungsfreundlichkeit und des allgemeinen Nutzens im Arbeitsalltag erarbeitet werden. Auf der Grundlage dieser Analyse soll zur Hilfenahme der Scriptsprache Java ein Tool entwickelt werden, welches After Effects um die eventuell fehlenden Funktionen erweitern soll.

Inhaltsverzeichnis

Abbildungsverzeichnis

Listingverzeichnis

1 Einleitung

1.1 Problemstellung

Grafiker, wie ich selbst, die in einer Postproduktionsfirma tätig sind und deren Aufgabenfeld vor allem in der Nachbearbeitung von Werbespots liegt, werden immer häufiger mit folgendem Problem konfrontiert: Ein Kunde möchte, bevor überhaupt ein Werbespot in die Produktion geht, verschiedene Ideen in einem Testlabor untersuchen. Da der finanzielle Aufwand für eine reale Produktion viel zu groß wäre, wird für jede Idee ein animiertes Storyboard erstellt. Diese nennen sich Animatics. Für mich als Grafiker heißt das, es werden für jede Einstellung Zeichnungen von einem professionellen Illustrator angefertigt, die dann von mir animiert werden müssen. Der Drang nach immer mehr Professionalität, bei gleichbleibender Arbeitszeit, gehört zum täglichen Geschäft. Vor zwei Jahren reichte es noch aus, die Illustrationen aneinander zu schneiden und kleine Kamerafahrten zu simulieren. Dagegen soll heute jedes erdenkliche Element animiert werden. Es fehlt oft die Zeit alles so zu produzieren, wie es der Kunde haben möchte. Besonders das Animieren von Personen ist aufwendig. Genau mit diesem Problem soll sich diese Arbeit auseinandersetzen. Auf Grundlage des sehr verbreiteten Animations- und Compositing Programms Adobe After Effects wird untersucht, inwieweit der Workflow speziell zum Animieren von 2D-Charakteren in Adobe After Effects optimiert werden kann.

1.2 Zielsetzung

Mit dieser Arbeit soll eine Lösung erarbeitet werden, die einem After-Effects-Benutzer ein Werkzeug zur Seite stellt, mit dessen Hilfe 2D-Charaktere in Adobe After Effects intuitiv und effizient animiert werden können. Um dieses Ziel zu erreichen, sollen die vorhanden Funktionen im Programm um ein selbst erarbeitetes Skript erweitert werden. Das Skript basiert auf der Programmiersprache JavaSkript, wobei spezielle Funktionen von Adobe für After Effects hinzugefügt wurden. In Erster Linie soll ein funktionsfähiges Modell erstellt werden, wobei schon während der Produktionsphase auf Benutzerfreundlichkeit und einfache Bedienung eingegangen wird.

1.3 Vorgehensweise

Im weiteren Verlauf werde ich zunächst auf den Ursprung der Charakteranimation und deren Grundlagen eingehen. Danach soll geklärt werden was Adobe After Effects aus macht und weshalb darin Charakteranimation möglich sein sollte. Außerdem werden die aktuellen Alternativen aufgezeigt und darauf aufbauend ein Pflichtenheft für das zu programmierende Skript erstellt. Danach wird mit der Dokumentation des Codes begonnen und als Abschluss das Resultat vorgestellt und unter animationstechnischen Gesichtspunkten ausgewertet.

2 Charakteranimation

2.1 Definition Animation

Animation ist vom lateinischen Wort „animare“ abgeleitet und bedeutet übersetzt „belebt“ oder „bewegt“. Damit wird die Abfolge von Bildern bezeichnet, die beim Ablauf eine ausreichend große Geschwindigkeit haben, dass für das menschliche Auge der Eindruck einer Bewegung entsteht.[1] Durch kleine Änderungen in den Einzelbildern gegenüber den Nachbarbildern wird ein flüssiger Übergang in den Bewegungen erwirkt. Die Einzelbilder können gezeichnet, im Computer berechnet, oder fotografische Aufnahmen sein.[2]

2.2 Arten der traditionellen Animation[3]

Legetrick

Bei dieser Art der Animation werden ausgeschnittene graphische Formen auf einen Tisch gelegt und dabei jeder Stand bildweise abfotografiert. Durch Verschieben der einzelnen Elemente wird so auf längere Sicht eine Bewegung entstehen.

Puppentrick

Beim Puppentrick kann man sagen, es ist die 3D Variante des Legetricks. Auch hier werden Puppen schrittweise von einer Kamera abgelichtet was dann eine flüssige Bewegung ergibt, auch Stopp-Trick (engl. Stop-motion) genannt.

Sachtrick

Diese Animationsform wird technisch, genau wie der Puppentrick durch schrittweises Abblichten der Miniaturszene umgesetzt. Unterschieden wird diese Art der Animation durch die darstellenden Elemente, denn hier sind Alltagsgegenstände wie Klammern, Eimer, Tassen usw. die Hauptakteure.

Zeichentrick

Der Zeichentrick ist wohl die bekannteste und populärste Art der Animation. In dieser Form werden zunächst die einzelnen Bewegungsabläufe durch Schlüsselbilder zeitlich festgehalten, und danach mit Zwischenbilder aufgefüllt. Dadurch ist ein effizienteres und durch die Spezialisierung der einzelnen Animatoren auch ein qualitativ hochwertigeres Ergebnis in kürzerer Zeit möglich.

1 Character Animation: 2D Skills for Better 3D, 2

2 Charakter-Animation in Film und Fernsehen, 8

3 Bewegung im Zeichentrick, 29f

2.3 Prinzipien der Animation

Die Animatoren von Walt Disney entwickelten in den 30er Jahren auf der Suche nach besseren und effizienteren Methoden Charaktere zum Leben zu erwecken die zwölf Prinzipien der Animation.[4] Diese spiegeln die Techniken wider, die gebraucht werden um lebendige Charaktere zu schaffen. Die Prinzipien gelten für fast alle Arten der Animation.[5]

- **Squash & Stretch (Quetschen & Strecken)**

Unter der Voraussetzung, dass das Volumen der Figur erhalten bleibt, kann diese in der Form gestaucht und gestreckt werden. So können die Auswirkungen von Geschwindigkeit, Trägheit, Gewicht und anderen äußeren Kräften dargestellt werden.

- **Anticipation (Ausholen oder Vorwegnehmen)**

Durch eine vorbereitende Bewegung wird der Betrachter auf den wesentlichen Teil des Bildes gelenkt. Wie beim Ausholen, bevor an den Fußball getreten wird, bewirkt die Gegenbewegung einen flüssigeren und natürlicheren Bewegungsablauf und schafft Aufmerksamkeit für den Betrachter.

- **Staging (Inszenierung der Posen)**

Die Szene muss so präsentiert werden, dass der Zuschauer eine unmissverständliche klare und eindeutige Aussage erhält. Dies beinhaltet den Ausdruck, die Persönlichkeit, eine Aktion oder eine Stimmung. Daneben trägt auch die Anordnung der Figuren innerhalb des Bildes zur Eindeutigkeit der Szene bei.

- **Straight Ahead & Pose-to-Pose**

Straight Ahead und Pose-to-Pose sind zwei unterschiedliche Animationstechniken. Für kreativere Bewegungsabläufe, wie bei einer Actionszene, wird häufig die Straight Ahead Technik angewandt. Dabei beginnt man beim ersten Bild der Szene und arbeitet sich bis an das Ende vor, wodurch der Animator bis auf die Hauptidee alle Freiheiten genießt, um ein dynamisches Ergebnis zu schaffen. Pose-to-Pose dagegen kann mit der Schlüsselbildmethode verglichen werden, denn hier werden von einem Animator Extremposen vorgezeichnet, die später durch andere Animatoren mit Zwischenbildern aufgefüllt werden. Dadurch hat man mehr Kontrolle über die Szene und kann durch die Spezialisierung auch schneller arbeiten.

4 Charakteranimation in Film und Fernsehen, 27

5 The Illusion of Life: Disney Animation, 47–69

- **Follow Through & Overlapping Action (weiterführende und überlappende Bewegung)**

Diese zwei Techniken sollen der Animation vor allem am Ende mehr Glaubwürdigkeit verleihen. Follow Through beschreibt dabei das Weiterführen einer Aktion, wobei kein abrupter Stop der Bewegung erfolgt, sondern sie darüber hinaus geht und danach langsam zum Stehen kommt. Auch die Animation von Haaren oder Stoff beruht auf diesem Prinzip. Overlapping Action erklärt, dass nicht alle Bewegungen gleichzeitig beginnen oder enden müssen. In der Praxis werden beide Prinzipien mit fließendem Übergang verwendet.

- **Slow In & Slow Out (Beschleunigung und Abbremsung)**

In der Natur sind konstante Geschwindigkeiten eher selten, sodass auch in der Animation das Prinzip von Beschleunigung und Abbremsung gilt. Daraus resultiert, dass am Anfang und am Ende einer Bewegung mehr Bilder gebraucht werden und zur Mitte immer weniger. Damit simuliert man eine langsame Beschleunigung und ein wieder Abbremsen, was auch ease-in und ease-out genannt wird.

- **Arcs (Bewegungsbögen)**

Aufgrund der Anatomie der meisten Figuren rotieren die Gliedmaßen häufig um ein Gelenk herum. Die Bewegungen sind deshalb eher bogenförmig und nicht linear. Daraus lassen sich besonders natürliche und weiche Bewegungen darstellen. Die Zwischenbildanimatoren haben die Aufgabe, diese fließenden Bewegungen zwischen den Schlüsselbildern zu generieren.

- **Secondary Action (zweitrangige oder unterstützende Bewegung)**

Eine sekundäre Handlung unterstützt und verstärkt die Haupthandlung. Dabei sollte die sekundäre Aktion nicht mit der Hauptaktion konkurrieren oder von ihr ablenken. Sie ist dazu da, um die Aussage, die Komplexität und die Glaubwürdigkeit der Szene zu erhöhen.

- **Timing (Bewegungsdauer)**

Timing beschreibt zum einen die Zeit, die für eine Aktion aufgewendet werden muss und zum anderen die Länge des Zeitraums zwischen einzelnen Aktionen. Die Zeitspanne, die für bestimmte Aktionen benötigt wird, kann die physikalische oder emotionale Bedeutung, je nach Länge der Bewegung variieren. Das Timing drückt die Persönlichkeit des Charakters durch seine Gestik, Mimik und deren Bewegungsabläufe aus.

- **Exaggeration (Übertreibung, Karikatur)**

Hierbei geht es wieder um ein Mittel, mit welchem der Betrachter auf die wesentlichen Elemente der Szene gelenkt wird. Das heißt, man sollte nicht willkürlich Aktionen überspitzen, da sie sonst zu hektisch und für den Zuschauer unglaubwürdig werden. Die Exaggeration kann sehr gut mit dem Squash-und-Stretch-Prinzip umgesetzt werden.

- **Solid Drawing (solides Zeichnen)**

Bezeichnet die Fähigkeit des Zeichners, Charaktere nicht nur aus allen Winkeln und in jeder erdenklichen Pose zeichnen zu können, sondern dies auch konstant in einer Animation aufrecht zu erhalten. Außerdem sollten die Charaktere lebendig sein und „wooden character“ vermieden werden, die beispielsweise parallel gezeichnete Beine besitzen, die dazu noch die gleiche Bewegung ausführen.

- **Appeal (Charisma, Charme und Reiz)**

Beschreibt den Gesamteindruck auf den Zuschauer, wobei nicht nur das Äußere, sondern auch die Posen und Bewegungen, den Charakter der Figur entsprechend interessant darstellen sollten. So kann auch ein böser Charakter hübsch sein, oder hässliche Figuren liebenswert. Es kommt darauf an, dem Zuschauer ein glaubhaftes Bild zu vermitteln, wobei die Figuren im Ganzen anziehend und spannend wirken sollten. Oft kann dies, wie oben beschrieben, durch eine Asymmetrie des Charakters erreicht werden.

2.4 Computeranimation

Unter Computeranimation versteht man die Berechnung von computergenerierten Bildern, die sich durch das jeweilige Vorgängerbild geringfügig unterscheiden und bei schneller Abfolge eine Bewegung darstellen. Häufig wird in der Computeranimation die Schlüsselbildanimation verwendet, dabei ändern sich nur einige Einzelbilder und deren Parameter wie Position, Form oder Rotation. Die für die Bewegung notwendigen Restbilder werden vom Computer durch Interpolation, auch Tweening genannt, ergänzt.[6]

Oft sind die zu animierenden Elemente hierarchisch angelegt, sodass die übergeordneten Objekte die unteren beeinflussen. Besonders Arme und Beine, die in der Charakteranimation eine große Rolle spielen, beinhalten große Parallelen zur Robotik. Hier sind die Strukturen auch an einem Stück modelliert und mit Gelenken verbunden. Durch die hierarchischen Abhängigkeiten dieses Aufbaus können direkte und indirekte Kinematik für die Charakteranimation angewandt werden. Für noch echtere Bewegungsabläufe kann zusätzlich das aufwändigere Motion-Capture-Verfahren hergenommen werden.

6 Computer Animation: Algorithms and Techniques, 2

2.4.1 Computeranimationstechniken

Keyframeanimation

In der Schlüsselbild- oder auch Keyframe-Animation wird für jedes zu animierende Parameter eines Objektes ein zeitlicher Punkt, also ein Keyframe im Animationsprogramm zugeordnet. Es erhält also jedes Parameter (Position, Rotation, Farbe, usw.) eines Elementes, welches eine zeitliche Änderung erfährt einen Eintrag auf der Zeitleiste. Die Parameter für die zwischen den Keyframes liegenden Zeitpunkte werden mit Hilfe mathematischer Interpolation von der Animationssoftware berechnet. Meistens ist dies die lineare-Interpolation oder die Kurven-Interpolation, die in Punkt 2.4.2 beschrieben wird.[7]

non-lineare Animation

Die non-lineare Animation ist nicht durch Zeit begrenzt, sondern agiert objektorientiert mit Animationsaktionen. Die verschiedene Aktionsclips, die individuelle Animationsabläufe enthalten, können unabhängig voneinander abgerufen werden. Dabei ist es möglich die Aktionen parallel und überschneidend ablaufen zu lassen. Als Beispiel kann man sich die Aktion „laufen“ und „winken“ vorstellen. Dabei kann der Charakter diese Aktionen nacheinander oder gleichzeitig ausführen.[8]

Bild-für-Bild-Animation

Bei dieser Form der Animation werden die fertigen Bilder nacheinander erstellt ohne Zwischenschritte zu absolvieren. Dies ist besonders bei komplexen Bewegungsabläufen hilfreich, denn hier kann bei größtmöglicher Kontrolle die Animation abgearbeitet werden.[9]

2.4.2 Interpolation

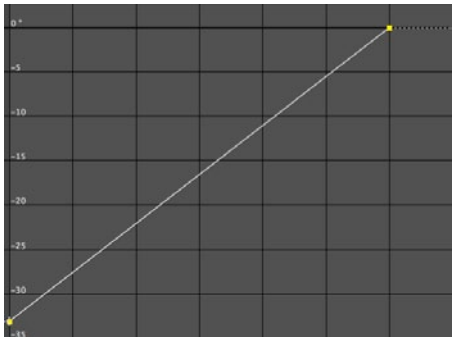


Abbildung 1: lineare Interpolation

Interpolation ist die Art und Weise wie der Computer die fehlenden Zwischenbilder, begrenzt durch die Schlüsselbilder, berechnet.

Die lineare Interpolation beschreibt die einfachste Form der Interpolation. Um die Zwischenbilder zu berechnen werden die Wertänderungen zwischen den Keyframes durch die Anzahl der noch fehlenden Bilder geteilt und auf die Zwischenbilder übertragen. Da-

7 Charakter-Animation in Film und Fernsehen, 91

8 http://softimage.wiki.softimage.com/xsidocs/nla_intro.htm#Rdv10200

9 Flash MX 2004 - Das offizielle Trainingsbuch, 164

durch entsteht eine konstante Geschwindigkeitsgerade, welche bei organischen Bewegungen äußerst mechanisch wirkt.

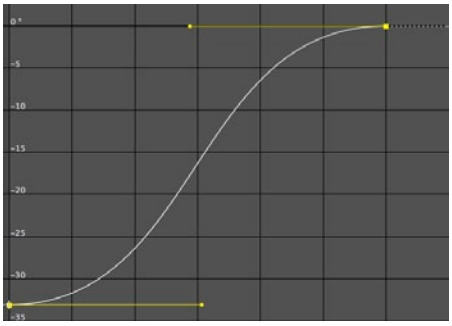


Abbildung 2: Kurven-Interpolation

Als Kurven-Interpolation bezeichnet man den zeitlichen Verlauf der Bewegung anhand einer Kurve, wodurch die Bewegung stärker beeinflusst werden kann. Dadurch sind Geschwindigkeitsänderungen umsetzbar, was Beschleunigung (engl. ease in) und Abbremsung (engl. ease out) ermöglicht. Die Zwischenbilder werden dabei als Kurve dargestellt, wobei ein starker Anstieg eine große Geschwindigkeit ergibt und ein flacher Anstieg eine geringe Geschwindigkeit.[10]

2.4.3 Rigging

Der Begriff „Rigging“ kommt aus der 3D-Animation und beschreibt eine dem Charakter untergeordnete Struktur, die das Animieren von diesem möglich macht. Der Charakter erhält eine Palette aus Bones (Knochen) und Joints (Gelenke), die wiederum von Kontrollobjekten bewegt werden können. Dabei ist die Animation immer nur so gut wie das zur Animation verwendete Rig. Es gibt immer verschiedene Möglichkeiten einen Charakter zu riggen, wozu einfache und komplexere Lösungen gehören. Meist orientiert man sich an einem tatsächlichen Skelettaufbau, der aus den Extremitäten und der verbindenden Wirbelsäule besteht. Es sollte vorher aber klar sein, in welcher Weise der Charakter genutzt wird und welche Herausforderungen daraus für das Rig entstehen.[11]

2.4.4 Kinematik

Kinematik ist die Lehre der Bewegung. Genauer gesagt ist es der Teil der Physik, welcher sich mit der Bewegung ohne Einwirkung von Kraft und Masse beschäftigt. Für die Charakteranimation gibt es zwei grundlegende Bewegungssysteme. Die direkte Kinematik (engl. forward kinematic) oder auch nur FK bezeichnet. Und die Inverse Kinematik (engl. Inverse kinematic) welche oft mit IK abgekürzt wird.

10 Charakter-Animation in Film und Fernsehen, 92

11 Body Language: Advanced 3D Character Rigging, 1

- **Direkte Kinematik**

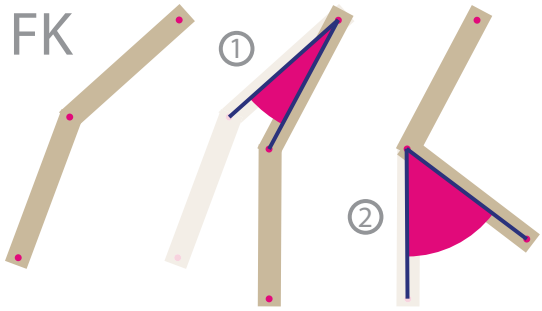


Abbildung 3: forward kinematik

Bei der direkten Kinematik sind die Gelenke oder auch Objekte in einer Hierarchie angeordnet, wobei die Animation vom obersten Objekt bis zum untersten erfolgt. Dies kann auch als „Eltern-Kind"-Beziehung betrachtet werden (engl. parent-child). Dabei ist beispielsweise der Oberarm das „parent"-Objekt des Unterarms und dieser wiederum hat die Hand als „child"-Objekt. Wenn nun die Hand bewegt wird und an ihr die Finger mit einer Kindbeziehung verankert sind, werden diese alle Transformationen der Hand nachahmen. Da der Unterarm der Vater der Hand ist, wird sich an dessen Position oder Rotation nichts ändern. Aufgrund der Tatsache, dass für jede Pose immer wieder von oben in der Hierarchie angefangen werden muss, ist diese Bewegungssteuerung mit viel manuellem Aufwand verbunden.[12]

- **Inverse Kinematik**

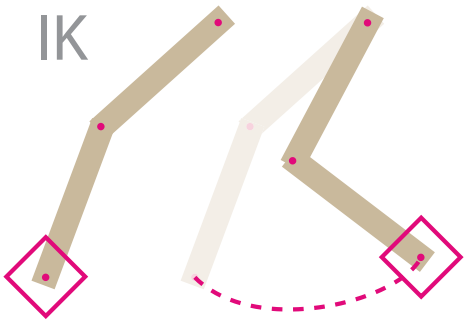


Abbildung 4: inverse kinematik

Das Gegenstück zur direkten Kinematik bildet die Inverse Kinematik. Die Struktur einer Hierarchie bleibt gleich, jedoch können Bewegungsabläufe bequemer realisiert werden. Die Transformationen werden vom untersten Gelenk der Hierarchie entlang nach oben weitergegeben. Es muss also nur das letzte Glied der sogenannten IK-Kette bewegt werden, wonach die anderen Glieder gezwungen sind sich so in ihrer Rotation zu ändern, dass die Position des Endeffektors eingenommen werden kann. Diese Animationsart ist intuitiver, da der Charakter eher wie eine Marionette bewegt wird. Es gibt immer mehrere Möglichkeiten um sich der gewünschte Position des letzten Gliedes anzugleichen, jedoch muss darauf geachtet werden, dass keine ungewollten oder anatomisch unmöglichen Posen entstehen. Dies kann einfach mit dem Begrenzen der Winkel umgesetzt werden.[13]

12 Charakter-Animation in Film und Fernsehen, 92-93

13 An Essential Introduction to Maya Character Rigging, 219-220

3 Compositing- und Motion-Graphics Programme

Compositing ist im Allgemeinen eine Technik, welche mindestens zwei unterschiedliche Bildelemente so zusammenfügt, dass die entstandene Einstellung für den Betrachter als stimmige Einheit erkennbar ist.[14] Motion-Graphics, auch Motion-Design genannt ist die audiovisuelle Gestaltung von Bewegtbild durch Typografie und Grafik-Design. Meist eingesetzt in Filmvorspännen, Trailern oder in der Werbung.[15] Es gibt zwei sehr verschiedene Compositingsysteme. Das erste, welches auch häufiger für Compositingaufgaben genutzt wird basiert, auf „Knoten". Wobei das zweite System, wozu auch Adobe After Effects gehört, auf Ebenen basiert und eher für Motion-Graphics genutzt wird.

3.1 Nodebasiert

Node kommt aus dem englischen und bedeutet Knoten. Jeder Knoten stellt ein Element, sei es ein Effekt, Ausgangsmaterial oder Resultat dar. Die Knoten sind untereinander verlinkt, sodass für den Anwender eine Baumstruktur entsteht die vom Ausgangsmaterial bis hin zum Resultat führt. Dieser Workflow hat den Vorteil, dass man auf einen Blick etwas ändern kann, was sofortige Auswirkungen auf das Resultat bewirkt. Es werden weniger Schlüsselbildanimationen benutzt, da der Workflow auf eine gesamte Lösung ausgelegt ist, beispielsweise zum Keying, oder Tracking. Vertreter für diesen Arbeitsablauf sind: Apple Shake, Fountry Nuke, Autodesk Flame, Toxic, Combustion, eyeon Fusion.[16]

3.2 Ebenenbasiert

In ebenenbasierten Compositingprogrammen wird jedes Medienelement in einer Komposition durch eine separate Ebene in der Zeitleiste dargestellt. Jede Ebene hat dabei eine eigene Länge, Schlüsselbilder und Effekte. Alle Ebenen liegen dabei aufeinander und können in ihrer Reihenfolge beliebig geändert werden. Um das finale Bild anzeigen zu können, wird der Ebenenstapel der Reihe nach, beginnend bei der untersten Ebene nach oben hin berechnet. Dadurch kann man sehr schnell 2D-Animationen und zum Teil auch einfache 3D-Animationen, wie sie bei Motion-Graphics eingesetzt werden, erstellen. Komplexere Compositingaufgaben wie das Keying und Tracking können in ebenenbasierten Programmen eine schwierige Aufgabe werden. Durch das einfache Verwenden von Schlüsselbildern ist diese Art von Compositingprogrammen sehr gut für Motion-Graphics geeignet. Der bekannteste Vertreter für diesen Arbeitsablauf ist Adobe After Effects.[17]

14 The Art and Science of Digital Compositing: Techniques for Visual Effects, Animation and Motion Graphics

15 http://www.mattfrantz.com/thesisandresearch/motiongraphics.html

16 http://mediablog.tmaekler.de/tag/node-based-compositing/

17 http://hig.diva-portal.org/smash/record.jsf?pid=diva2:327273

4 Adobe After Effects

4.1 Definition

Adobe After Effects ist ein Animations- und Compositingprodukt des Softwareherstellers Adobe Systems. Es bietet die Möglichkeit, Bild und Ton, Video, Typografie und 3D-Animationen miteinander zu kombinieren und in den verschiedensten Formaten zu exportieren.[18] Nicht nur das Zusammenfügen von Bildmaterial gehört zum Umfang des Programms, sondern auch das Animieren von Formen und Effekten. Nicht zuletzt wegen der komfortablen Zeitleiste ist es möglich, nahezu jeden Parameter durch Schlüsselbilder (keyframes) animieren zu können. Durch die weitreichende Funktionalität des Programms und die Benutzerfreundlichkeit bedient After Effects eine große Zielgruppe, angefangen beim Hobbyfilmer bis hin zum Profianwender.

4.2 Aufgabenfeld

After Effects wird von Motion-Designern und Compositoren gleichermaßen genutzt. Für Filmschaffende, Mediengestalter und Webdesigner ist es das Werkzeug der Wahl.[19] So werden mit dem Programm Logoanimationen, Sendedesigns und Präsentationen erstellt, sowie auch Retuschen, Rotoscoping, Tracking und Keying.

4.3 Aufbau und Funktionsweise

Adobe After Effects ist in einzelne Paletten unterteilt, die seit Version sieben andockbar sind, das heißt es ist möglich den Arbeitsbereich den persönlichen Bedürfnissen anzupassen.[20] Die Software besteht dabei aus drei Hauptpaletten. Den Anfange macht die Zeitleiste, hier wird das Ausgangsmaterial bearbeitet und mit Effekten und Schlüsselbildern (Keyframes) versehen. Die Viewportpalette ist in der Hinsicht wichtig, da sie das Endresultat abbildet. Es können sofort die Auswirkungen der hinzugefügten Effekte betrachtet werden. Außerdem ist es möglich Masken direkt auf dem dargestellten Material zu zeichnen und verschiedene Plugins zu bedienen. Als dritte essentielle Palette kann die Projektpallette angesehen werden, in dieser findet man alle Ausgangsmaterialien vom Audiofile, über Farbflächen und Filmclips bis hin zu Kompositionen. Natürlich gibt es noch weit mehr Paletten, zum Beispiel für Audiopegel, Tracking, Effekte und Schriftarteinstellungen.

Eine Komposition ist im weitesten Sinne wie ein neues Blatt Papier. Man beginnt mit den Einstellungen für Auflösung und Bildrate und kann anfangen Quellmaterial zusammenzuführen und mit Effekten zu versehen. Um den Überblick zu wahren und das

18 Adobe After Effects CS3, Das Praxisbuch zum Lernen und Nachschlagen, 29

19 Adobe After Effects CS3, Das Praxisbuch zum Lernen und Nachschlagen, 29

20 Creating Motion Graphics with After Effects, 4

Arbeiten einfacher zu gestalten ist es möglich und oft auch nötig, Kompositionen ineinander zu verschachteln, sodass eigene separate Einheiten entstehen, die wiederum mit Masken und Effekten bearbeitet werden können.

5 Analyse der aktuellen Situation

5.1 Verwendung von Charakteranimation in Adobe After Effects

Adobe After Effects ist ein sehr weit verbreitetes Programm, das von Postproduktionen, Werbeagenturen und anderen grafischen Einrichtungen gleichermaßen genutzt wird, um animierte Bewegtbilder zu erstellen. Die Vorteile liegen in dem zeitleistenbezogenen Arbeitsablauf, der Kompatibilität zu anderen Adobeprodukten, den sehr mächtigen Animationstools, Expressions, Masken und dem sehr leichten Einstieg.

Charakteranimation kann bisher in After Effects nur auf einem Minimum betrieben werden, obwohl die Anforderungen immer vielfältiger werden. Besonders bei Animatics, animierten Storyboards, welche sehr oft in Adobe After Effects produziert werden, würde ein Charaktertool sehr hilfreich sein. Die Zeiten in denen es dem Kunden reichte, ein paar Ebenen von A nach B zu verschieben und dabei die Kamera etwas zu zoomen sind vorbei. Wie in allen anderen Teilen der Branche wird auch hier der Anspruch immer höher. So sollen beispielsweise die einzelnen Figuren richtig laufen und gehen können, weshalb in den von Illustratoren angelieferten Photoshopdateien jedes animierbare Objekt als separate Ebene vorliegt. Das Aufsetzen eines funktionstüchtigen Setups braucht jedoch Zeit, die normalerweise nicht vorhanden ist, denn für ein Animatic sind im Durchschnitt zwei Tage Arbeit (Änderungen eingeschlossen) eingeplant. Durch das Puppettool konnten schon kleinere Animationen, wie das Heben eines Arms sehr schnell und zeitsparend animiert werden, jedoch sind komplizierte Animationen, wie laufende Charaktere aufgrund eines fehlenden Bonesystems und einer Inversen Kinematik sehr schwierig.

Wenn die Qualität von 2D-Charakteranimationsprogrammen erreicht werden soll, stellt sich die Animation einer Werbefigur als große Herausforderung dar. Es muss jedoch festgehalten werden, dass es nicht das Ziel ist mit Adobe After Effects professionelle Zeichentrickfilme zu produzieren, denn dafür gibt es spezielle Programme. Vielmehr soll ein Workflow erarbeitet werden, um einfache Charakteranimationen zeitsparnend für Motion-Graphicer und Motion-Designer zu ermöglichen.

5.2 Animationsmöglichkeiten in Adobe After Effects

Adobe After Effects hat im Moment die Version 10 mit der Typbezeichnung CS5.5 erreicht. An den letzten Verbesserungen ist sehr gut zu erkennen, dass der Hersteller Adobe sein Produkt im Compositingmarkt besser darstellen möchte. Im jüngsten Update sieht man das sehr deutlich.

- 64Bit Anbindung
- Rotobrush (eine Hilfe zum Rotoscopieren)
- importieren von AVC-Intra und RED-Material

- Verbesserung des internen Farbkorrekturplugins (Colorfiness)
- Verbesserung des Planartrackers Mocha
- Unterstützung von Lookup-Tabellen (LUT) für Farben[21]

Das Puppettool, welches mit der Version CS3 eingeführt wurde war das letzte Feature, das für Motion-Graphicer in Bezug auf Charakteranimation interessant war. Im Endeffekt war es jedoch ein animierbarer Deformer, der Verzerrungen durch positionierbare Knoten ermöglicht.[22] Das Puppettool beinhaltet keine Funktionen der Charakteranimation, weder ein Bonesystem noch Inverse Kinematik oder ein Animationsrig. Dennoch kann After Effects durch das Parentingsystem eine Direkte-Kinematik-Kette erstellen. Aber wie schon im Punkt 2.4.4 beschrieben, ist der Aufwand dieses System zu animieren sehr hoch.

5.3 Expressions und Skripte

In Adobe After Effects ist es seit Version 7 möglich den Arbeitsablauf mit eigenen Skripten zu erweitern. Die Skripte werden dabei in Javascript mit dem Standard ECMA-262[23] geschrieben und haben die Endung jsx. Sie können jederzeit aufgerufen werden und sind in der Lage, fast alle Eigenschaften zu automatisieren und zu manipulieren. Dazu zählt das Erstellen von ganzen Kompositionen mit den dazugehörigen Elementen. Die Kontrolle über das Ausgabemodul und sogar Kommunikation mit dem Internet ist möglich. Außerdem verfügt die Skriptsprache über eine Bibliothek um eigene Benutzeroberflächen zu gestalten, die wiederum den Arbeitsfluss und die Integration in After Effects verstärken.

Auf der anderen Seite stehen die Expressions, welche auch auf Javascript basieren, aber direkt in After Effects geschrieben werden und kein externes Programm nötig ist. Jede Expression wird direkt in die Eigenschaft geschrieben, welche durch den Codeabschnitt verändert werden soll (Position, Rotation, Skalierung, oder ein Effekt). Diese wirken immer sofort und bringen mitunter Effekte zustande, die anders nicht möglich wären.[24]

Der Hauptunterschied zwischen den beiden Javascriptformen liegt in der Anwendung und Ausführung. Expressions werden immer ausgeführt, quasi verändern sie live die Eigenschaften der Ebenen und deren Elemente. Skripte dagegen brauchen ein Startsignal und arbeiten dann eine Folge von Befehlen ab. Man kann also daraus schließen, dass Skripte nicht mit Schlüsselbildern versehen werden können, Expressions dage-

21 http://www.adobe.com/de/products/aftereffects/buying-guide.html
22 http://www.adobe.com/de/designcenter/aftereffects/articles/aftcs3it_ciblesson8.html
23 Adobe After Effects CS3 Professional SCRIPTING GUIDE, 9
24 After Effects Expressions, XI

gen schon. Um Adobe After Effects nun neue Features hinzuzufügen, ohne ein komplettes Plugin zu schreiben, können beide Varianten miteinander verbunden werden.[25]

5.4 Bestehende Charakteranimations-Skripte /-Expressions

5.4.1 Dan Ebbert

Dan Ebbert ist der Experte wenn es um Skripte und Expressions in After Effects geht. Auf seiner Seite (http://www.motionscript.com) können anhand von Beispielen und dokumentierten Codeabschnitten die Grundlagen erlernt werden. Neben der Einführung in die Materie beschäftigt er sich auch mit der Umsetzung von neuen Features, die so in After Effekts noch nicht existieren. Darunter fällt auch eine Expression, die Inverse Kinematik mit einem Gelenk ermöglicht.[26] Für Normalanwender ist diese Expression nur schwer handhabbar, da diverse Codeanpassungen vorgenommen werden müssen bevor die Expression für das eigene Setup funktioniert. In Erster Linie stellt dies auch nur einen Prototyp dar, der zeigen soll, dass die Basis für komfortable Charakteranimation in After Effects möglich ist.

5.4.2 Duik Tools: DuDuF IK Tools for After Effects

Der zweite, der auch das Potential in dieser Richtung erkannte war Nicolas Dufresne, ein französischer Motion-Graphic-Designer und Skriptschreiber. Auf der Grundlage von Dan Ebberts Basisexpression schrieb er ein Skript für After Effects, welches per Mausklick eine IK-Kette zwischen zwei ausgewählte Ebenen legte. In den darauf folgenden Versionen des Skriptes wurden immer mehr Funktionen hinzugefügt. In der Aktuellen ist es nun möglich eine IK-Kette aus Puppettoolpunkten zu generieren, was dazu führt, dass der 2D-Charakter aufgrund der Verzerrungseigenschaften des Puppettools organischer bewegt werden kann.[27]

5.5 Alternative 2D Animationssoftware

5.5.1 Toon Boom Animation Inc.

Der Marktführer in 2D Zeichentrickanimationssoftware ist Toon Boom Animation inc. Zu den Nutzern zählen unter anderem Walt Disney Animation Studios und Walt Disney Television Animation, Nelvana, Warner Bros. Animation. Die Features erstrecken sich von Inverser Kinematik über 3D Kameras und Ebenen bis hin zu Morphtargets und Lippensynchronisation. Es wird in verschiedenen Ausstattungen vertrieben, welche sich

25 http://www.motionSkript.com/ae-Skripting/introduction.html
26 http://www.motionSkript.com/expressions-lab-ae65/ik.html
27 http://www.duduf.com/

stark in Preis und Komplexität unterscheiden[28]

5.5.2 Retas Studio

Retas (Revolutionary Engineering Total Animation System) ist ein 2D Animationsprogramm, welches für Microsoft Windows und Apple Mac OS erschienen ist. Vor allem in Japan kommt es bei der Produktion von Animes (japanische Zeichentrickfilme) zum Einsatz und gehört dort zu den Marktführern. Das Programm kann man in verschiedenen Ausstattungen erwerben, wobei die Hauptfeatures ähnlich wie bei Toon Boom sind. Zu den Funktionen der Software gehört ein Layersystem, Bones mit Inverser Kinematik, virtuelle Kameras und Export der Animation in Adobe Flash Format oder Apple Quicktime.[29]

5.5.3 Anime Studio

Anime Studio vom Entwickler Smith Micro Software ist eine Alternative für Normalverbraucher. Es bietet ähnlich viele Funktionen wie die vorher genannten Programme und beinhaltet sogar eine einfache Physik-Engine. Die Software ist eher auf das Animieren mit der Tweening-Methode ausgerichtet und weniger auf das Zeichnen von einzelnen Bildern.[30]

5.5.4 Adobe Flash

Adobe Flash ist sehr verbreitet wenn es um die Animation von Webinhalten geht. Flash und auch die obigen Programme sind größtenteils dazu übergegangen verlustfreie Vectorgrafiken zu animieren. Für Animationen im Internet ist dies die Voraussetzung um wenig Speicher zu verbrauchen, da sie nicht Pixel für Pixel abgerufen werden müssen, sondern durch mathematische Funktionen beschrieben werden können. Außerdem sind damit sehr einfach spezielle Effekte zu realisieren, wie Mophings, Kollisionsabfragen und verschiedene Outlineeffekte. Flash selbst hat hier den Grundstein gelegt und beinhaltet ebenso ein Bonesystem als auch eine der stärksten Skriptsprachen mit der sich beispielsweise auch Inverse Kinematik programmieren lässt.[31]

28 http://beta.toonboom.com/professionals/animate-pro/features
29 http://www.ex.org/4.2/09-column_bts1.html
30 http://anime.smithmicro.com/asp-features.html
31 http://www.adobe.com/de/products/flash/features.html

5.6 Zusammenfassung

Adobe After Effekts ist mit den gegebenen Werkzeugen nicht ausreichend ausgestattet, um als 2D-Charakteranimationsprogramm nutzbar zu sein. Lediglich das Puppet-Tool, welches in Version CS3 eingeführt wurde beinhaltet die ersten Ansätze in dieser Richtung.

Abhilfe können Skripte und Expressions schaffen, wie die Inverse Kinematik Expression von Dan Ebbert. Dieser Workaround ist jedoch noch nicht ausreichend, um in einer professionellen Arbeitsumgebung eingesetzt zu werden. Der Aufwand, der betrieben werden muss um die Expression an die eigenen Bedürfnisse anzugleichen, ist verhältnismäßig hoch und verbraucht wertvolle Zeit. Eine automatisierte Variante, die durch ein Skript bewerkstelligt wird, erarbeitete Nicolas Dufresne mit seinen „DuDuF IK-Tools“. Er hatte das Problem erkannt und nahm die grundlegende IK-Expression von Dan Ebbert und verpackte sie in einem Skript. Die Anwendungsfreundlichkeit ist enorm gestiegen und das Skipt wird auch in der After Effects Community sehr gut angenommen. Eine IK-Kette die auch mit dem Puppettool funktioniert gehört zu den neueren Features des „DuDuF IK-Tools“. Es fehlen aber dennoch einige essentielle Funktionen, wie die eines Abrollmechanismus und die einer Gesamtlösung um Charaktere zu riggen und zu animieren.

Als Alternative könnten auch die zahlreichen professionellen 2D-Charakteranimaitonsprogramme für solche Aufgaben genutzt werden, wobei der Kosten-Nutzen-Faktor eher gering ist. Da diese Programme auch für einen professionellen Workflow ausgelegt sind, ist deren Komplexität nicht geringer als die von Adobe After Effects. Demzufolge wird auch eine gewisse Einarbeitungszeit benötigt, um die verlangten Aufgaben umzusetzen. Ein weiteres Manko ist die Kompatibilität zwischen den Programmen. Jede Charakteranimation müsste aus dem Programm exportiert werden, um sie dann in das bestehende Setup in After Effects einzubinden. Treten hier wieder Änderungen auf, muss der Prozess ständig wiederholt werden. Auch Adobe Flash bietet noch keinen zufriedenstellenden Austausch von Projekteinstellungen an, der einen flüssigeren Arbeitsablauf garantiert.

5.7 Schlussfolgerung

Nach der Betrachtung der vorhandenen Möglichkeiten, 2D-Charakteranimation in Adobe After Effects mit einem effektiven Arbeitsablauf zu kombinieren, muss nun erörtert werden, welche Punkte nötig sind um dies zu erreichen.

- Workflow ohne dritte Programme
- Nutzung von Expressions und Skripten
- Benutzerfreundlichkeit

- Schnell und Effizient
- Einfache Animation mit Kontrollobjekten
- Automatisierte Erstellung des Charakter-Rigs
- Anpassbar
- Inverse Kinematik
- Abrollmechanismus

Unter diesen Punkten soll eine geeignete Lösung gefunden werden. After Effects kann im Funktionsumfang nur durch Plugins, Skripte und Expressions erweitert werden. Da Plugins einen beträchtlichen Mehraufwand bedeuten, soll für After Effects ein Skript erarbeitet werden, das alle Punkte so gut es geht in sich vereint.

Bevor mit dem Code-Schreiben angefangen wird, muss zunächst geklärt werden inwieweit sich Arbeitsabläufe automatisieren lassen. Dafür wird in After Effects als Erstes ein Charakter-Rig manuell erstellt und darauf geachtet, welche Hürden und Schwierigkeiten entstehen. Ein zweibeiniger Charakter soll dabei aus mindestens zwei Füßen mit dazugehörigen Zehen, aus zwei Ober- und Unterschenkeln, zwei Ober- und Unterarmen, zwei Händen mit jeweils einer Gruppe Fingern, einem Ober- und Unterkörper, Hals und Kopf bestehen. Dies sind jedoch keine Begrenzungen, sondern die Basis um die essentiellen Kontrollobjekte zu etablieren. Zum Beispiel für die Inverse Kinematik, die Abrollmechanik und andere Kontrollmechanismen. Außerdem wird nur mit einer vorgegebenen Basis eine Automation des Charakter-Rigs möglich.

Während der Entwicklung des ersten Prototyps fallen markante Arbeitsschritte auf, die für das spätere Skript übernehmbar sind. Ausgehend von einem Charakter, der in Photoshop mit den oben genannten Einzelteilen gestaltet wurde, müssen als Erstes die Gelenke für jedes Objekt festgelegt werden. Wenn alle Gelenke gesetzt sind, kann der Charakter, mit den Füßen beginnend, zusammengesetzt werden. Da After Effects in der Lage ist die Größen der einzelnen Teilen auszulesen, kann mit der Hilfe von Positions- und Gelenkwerten (Ankerpunkte) auch dieser Schritt in das Skript übertragen werden. Nun müssen die Elemente miteinander durch das Parenting-System von After Effects verbunden werden, sodass eine hierarchische Struktur zwischen den Körperteilen entsteht. Jetzt fehlen noch die Expressions, die für den Abrollmechanismus und für die Inverse Kinematik zuständig sind und das komplette Rig animierbar machen. Eine detaillierte Dokumentation der Funktionsweise wird in Abschnitt 7 erfolgen.

6 Pflichtenheft

6.1 Ausgangssituation

Auch vom Motion-Graphic-Designer wird immer häufiger verlangt, einfache Charakteranimationen zu produzieren. Professionelle 2D-Charakteranimationssoftware wäre für diesen Einsatz einfach zu umfangreich und aufwendig. Mit Dan Ebberts und Nicolas Dufresnes Expressions beziehungsweise Skripten gibt es bisher zwei Ansätze um innerhalb von Adobe After Effects Charakteranimation komfortabel umzusetzen. Diese Ausgangssituation beachtend soll ein Skript mit den dazugehörigen Expressions entwickelt werden, welches mehr Features, eine benutzerfreundliche Bedienung und schnelles Arbeiten in sich vereint.

6.2 Zielsetzung

Das 2D-Rigging Skript soll die Funktionalität von Adobe After Effects um eine schnelle und effizientere Funktion zum Animieren und Riggen von 2D-Charakteren erweitern.

Musskriterien:

- intuitiv, keine große Einarbeitungszeit nötig
- Stabilität
- Möglichkeit Verbesserungen nach dem Rigging vorzunehmen
- dynamischer Charakteraufbau

Sollkriterien:

- Fersen- und Ballenfunktion
- Mehrsprachigkeit
- Erweiterbarkeit

Kannkriterien:

- Unterstützung des Puppettools
- Rigging von Vielbeinern

Abgrenzungskriterien:

- keine Kopie der Inversen Kinematikexpression von Dan Ebbert

6.3 Produkteinsatz

Das Skript kann von jedem genutzt werden, der sich mit Charakteranimation in Adobe After Effects auseinandersetzen möchte.

6.4 Anwendungsbereich

Skript zur Vorbereitung eines animierbaren 2D-Charakters in Adobe After Effects mit den dazugehörigen Animationshilfsmitteln.

6.5 Zielgruppe

Die Applikation ist an alle Motion-Graphic-Artists gerichtet, die sonst ein anderes Programm für Charakteranimation genutzt haben und im schnelllebigen Berufsalltag eine Alternative austesten wollen. Auch Nutzer von anderen Skripte, können es für ihren Arbeitsablauf nutzen und damit ihre Möglichkeiten erweitern. Voraussetzung ist ein sicherer Umgang mit Adobe After Effects.

6.6 Betriebsbedingungen

Das Skript funktioniert ab Adobe After Effects CS3 und wird die Sprachversionen Englisch, Deutsch, Französisch, Spanisch und Italienisch unterstützen.

6.7 Geplante Features

Das Hauptaugenmerk soll auf Benutzerfreundlichkeit und Schnelligkeit liegen. Ausgehend von diesen zwei Grundvoraussetzungen soll der Programmablauf entwickelt werden. Benutzerfreundlichkeit wird erreicht indem viele Arbeitsschritte automatisiert und vereinfacht werden, sodass die gegebenen Funktionen klar ersichtlich sind und Nutzungsfehler von vornherein abgefangen werden können. Viele Fehlerquellen werden mit einem Vier-Punkte-Workflow vermieden, welcher benutzerfreundlich und logisch ist. Aber dennoch so viel Freiheit und Schnelligkeit bietet, dass ein effizientes Arbeiten möglich ist.

Aufbau des Charakters

Um den Vier-Phasen-Ablauf nachzuvollziehen muss zunächst der Aufbau des endgültigen Rigs geklärt werden. Die Ausgangssituation für die wichtigsten Expressions benötigt ein Minimum an Körperteilen, die vom Nutzer in der Skriptbenutzeroberfläche angegeben werden müssen. Folgende Elemente muss der Charaker besitzen damit, die Inverse Kinematik, der Abrollmechanismus und andere Rotationsexpressions funktionieren: Kopf, Hals, zwei Körperteile, Oberarm, Unterarm, Hand, Finger, Oberschenkel, Unterschenkel, Fuß und Zeh. Dabei muss von Objekten, bei denen eine linke und

rechte Ausführung vorhanden ist, nur eine davon erstellt werden. Der Gegenpart dazu wird innerhalb des Skriptablaufs aus den Bestehenden generiert. Um die Freiheit des Nutzers nicht zu stark einzugrenzen, können nicht benötigte Elemente durch Null-Objekte in After Effects ersetzt werden. Natürlich besteht ebenso die Möglichkeit Accessoires oder andere Teile nach dem Rigging am Charakter nachträglich anzufügen oder komplette Elemente einfach auszutauschen.

6.8 Grobablauf

Das Skript gliedert sich in vier Phasen. Die erste erfasst die vom Programm gegebenen Ebenen mit Körperteilen. In der zweiten Phase werden Ankerpunkte für die Gelenke gesetzt, welche die Grundlage der dritten Phase bilden. In dieser wird der Charakter dynamisch, basierend auf den Maßen seiner Körperteile und der Platzierung der Ankerpunkte positioniert. In der letzten Phase werden schließlich alle Abhängigkeiten und Expressions, die zum Animieren nötig sind, hinzugefügt und die komplette Komposition aufgeräumt.

6.8.1 Ebenen laden

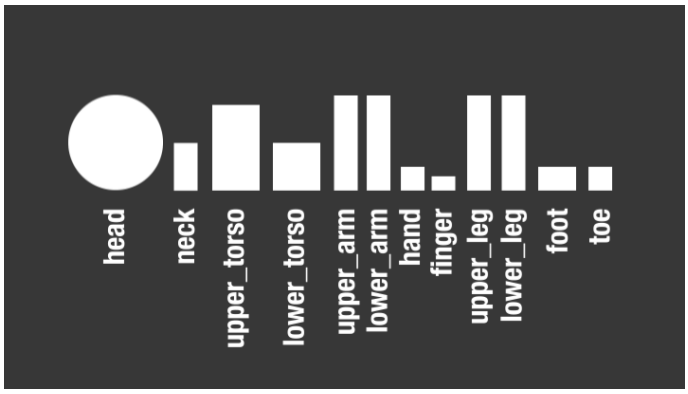


Abbildung 5: Ebenen laden

Teilablauf Ebenen laden

Um den ersten Schritt auszuführen muss eine Komposition erstellt werden, welche exakt 12 Ebenen beinhaltet. Mit ausgewählter Komposition klickt man nun auf den „Ebenen laden“ Button und die einzelnen Körperteile werden in das Skriptfenster übertragen. Bevor mit dem nächsten Abschnitt weitergemacht wird, müssen die Ebenen im Skriptfenster ihrem zugehörigen Slot zugewiesen werden.

Vorgang Ebenen laden

Nachdem das Userinterface erstellt und von After Effects ausgeführt wird, können die zwölf Charakterteile in das Skript geladen werden. Zunächst muss geprüft werden ob

eine Komposition ausgewählt wurde, nur dann kann das Skript die darin enthaltenen Ebenen erkennen und für sich zur Weiterverarbeitung speichern. Falls keine Komposition ausgewählt wurde und man dennoch auf „Ebenen laden“ klickt, wird das Skript eine Fehlermeldung ausgeben.

Ist diese Hürde genommen, werden nun, die in der ausgewählten Komposition enthaltenen Ebenen, in ein Array[32] geschrieben, welches von einem Dropdownmenü ausgelesen wird. Jedem Slot wird das gleiche Dropdownmenü zugeordnet, wobei die Slots ihren eigenen Standardplatz im Array haben. Besteht der Wunsch den Workflow zu beschleunigen, müssen die Charakterteile vor dem Laden der Ebenen schon in die Reihenfolge gebracht werden, die vom Skript standardmäßig angenommen wird. Andernfalls wird als Nächstes für jeden Slot der richtige Part im Dropdownmenü ausgewählt. Falls der zu animierende Charakter nicht aus genau zwölf Elementen besteht, sollten nicht verwendete Teile dennoch mit einem Nullobjekt, welches nur als Platzhalter dient, ersetzt werden. Es ist nur mit genau zwölf Elementen eine fehlerfreie Erstellung der Figur gegeben.

Dialog Ebenen laden

Als Bildschirmausgabe werden die einzelnen Elemente im Skriptfenster in die dazugehörigen Dropdownmenüs geschrieben.

6.8.2 Ankerpunkte setzen

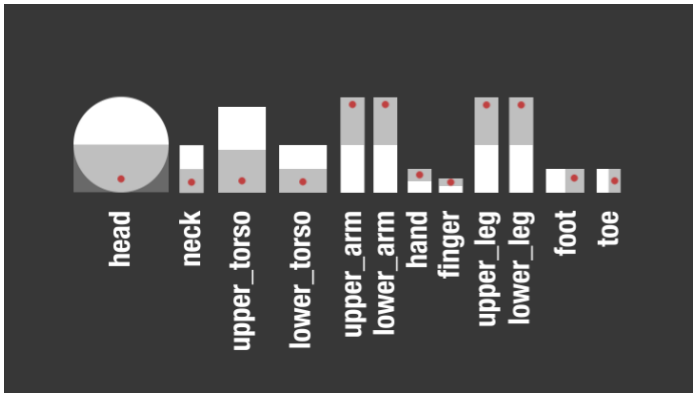


Abbildung 6: Ankerpunkte setzen

Teilablauf Ankerpunkte setzen

Sobald alle Körperteile ihrem zugehörigen Slot zugewiesen wurden, kann man mit dem Drücken des „Ankerpunkte setzen“-Buttons fortfahren. Es werden nun Hilfspunkte generiert mit denen bestimmt wird, wo die jeweiligen Ankerpunkte beziehungsweise Gelenke der Körperteile liegen sollen.

32 Variable mit mehreren Einträgen

Vorgang Ankerpunkte setzen

Zunächst muss im Skript ausgewählt werden welche Ausrichtung die zu riggende Figur haben soll. Im Skriptfenster sind dafür drei Radiobuttons vorgesehen. Je einer für eine linke, rechte und frontale Ausrichtung. Wählt man hier nichts aus, gibt das Skript beim klicken auf „Ankerpunkte setzen“ einen Fehler aus. Das Skript selbst generiert als Erstes an den aktuellen Positionen jedes einzelnen Elements eine rote Farbfläche mit den Abmessungen, die im Feld Dummy-Größe vorher bestimmt werden können. Zusätzlich wird eine Maske auf die viereckige Farbfläche gelegt, sodass Punkte zum Verschieben und Positionieren entstehen. Vorbereitend für den Aufbaumechanismus, also wenn die Figur im nächsten Schritt zusammengesetzt wird, müssen die Punkte in einer vorgegebenen Zone des Körperteils geschoben werden. Zur Orientierung und Hilfestellung für den Nutzer erstellt das Skript für jedes Charakterelement halbtransparente Farbflächen, welche die Zone markieren in der der Ankerpunktdummy positioniert werden kann. Damit nur die Dummys bewegt werden können, sind die restlichen Ebenen geschlossen und für den Benutzer unantastbar.

Dialog Ankerpunkte setzen

Das Skriptfenster selbst gibt keine Ereignisse zurück, die den Erfolg des Schrittes bestätigen. Dies ist jedoch erkennbar wenn kein Köperteil bewegt werden kann, aber dafür jedes einen roten Punkt und eine halbtransparente Fläche besitzt.

6.8.3 Charakter aufbauen

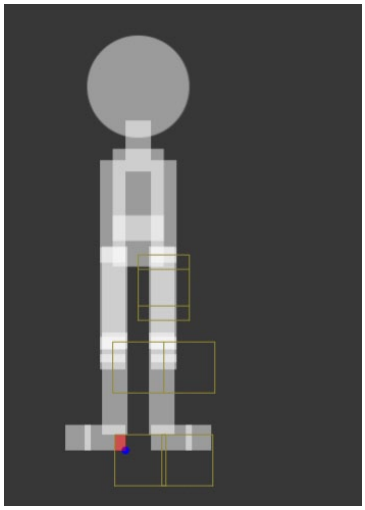


Abbildung 7: Charakter aufbauen

Teilablauf Charakter aufbauen

Wenn alle Ankerpunkte gesetzt wurden und auch innerhalb der dazugehörigen Zonen liegen kann mit dem „Charakter aufbauen“ Button fortgefahren werden. Sobald der Knopf gedrückt wird, werden alle Hilfsflächen sowie Dummys gelöscht. Anhand der Größen der einzelnen Körperteile und der Position der Ankerpunkte wird der Charakter von unten nach oben aufgebaut.

Vorgang Charakter aufbauen

Im Skript werden zunächst alle Positionen der Dummys an die Ankerpunktpositionen der Körperteile übertragen. Die Hilfselemente werden nun nicht mehr gebraucht und somit gelöscht. Von den Zehen ausgehend wird nun der Charakter aufgebaut. Durch die

Berechnungen im Skript werden die einzelnen Elemente während des Aufbaus überlappend aneinandergesetzt, sodass sich die Ankerpunkte annähernd da befinden wo sie der Nutzer erwartet. Außerdem erstellt das Skript verschiedene Nullobjekte die anhand der Positionierungsdaten am Charakter ausgerichtet werden. Diese werden später für die Animation des Charakters nötig sein. Als Letztes kann der Benutzer in diesem Stadium die Möglichkeit wahrnehmen, Änderungen an der Positionierung vorzunehmen und Elemente nach seinen Wünschen zu verschieben.

Dialog Charakter aufbauen

Als Ausgabe wird man bei erfolgreicher Ausführung einen Charakter sehen, welcher mit zusätzlichen Nullobjekten bestückt ist. Falls rote Punkte oder Hilfsflächen in der Komposition vorhanden sind, war die Ausführung nicht erfolgreich.

6.8.4 Charakterteile verbinden

Abbildung 8: Charakterteile verbinden

Teilablauf Charakterteile verbinden

Wenn nun alle Teile so angebracht sind, dass sie den Vorstellungen des Users entsprechen, kann der letzte Schritt gestartet werden. Mit dem Klick auf „Link“ werden nun alle Elemente untereinander verbunden und mit Expressions bestückt. Außerdem werden Effekte zur Animation eingefügt und zum Schluss die Zeitleiste aufgeräumt, sodass nur noch animationsrelevante Elemente sichtbar und bearbeitbar bleiben.

Vorgang Charakterteile verbinden

Als Erstes werden im Skript alle Objekte untereinander abhängig gemacht, wobei eine hierarchische Eltern-Kind-Beziehung entsteht. Dann werden Effekte für die zu steuernden Expressions und die dazugehörigen Expressioncodes erstellt. Wenn alles an die Ebenen übertragen wurde, werden die nicht für die Animation relevanten Objekte für den Nutzer gesperrt und in der Zeitleiste ausgeblendet.

Dialog Charakterteile verbinden

Das Skriptfenster selbst zeigt nichts an, um den Erfolg der Prozedur zu bestätigen. Wenn in der After Effects-Zeitleiste nur noch fünf Nullobjekte übrig sind, war dieser Schritt erfolgreich.

7 Entwicklung des Skripts

Der nachfolgende Abschnitt wird das aktuelle Skript dokumentieren. Dabei wird auf grundlegende Sachverhalte detailliert eingegangen. Aufgrund der Übersicht werden nur die allgemeinen Funktionsweisen des Skriptes erläutert und sich wiederholende Abschnitte übersprungen.

7.1 Initialisierung

```
function Rig() //Hauptfunktion
{
          $.writeln(„START“);
          var Rig = this;
          var desc = „RigIt - character rigging for After Effects“;

          if (parseFloat(app.version) < 7)
                    {
                              alert(„This script requires After Effects 7 or later.“, „RIgIt“);
                              return;
                    }
                    else
{

[...]

new Rig().run(this); //Start und Speicherreservierung
```

Listing 1: Main-Funktion

Damit After Effects überhaupt das Skript ausführt, wird eine Hauptfunktion, in Listing 1 „Rig“ genannt, deklariert. Diese beinhaltet den gesamten Code des Programms. Um die Funktion jedoch zu starten ist ein separater Befehl nötig, der nach der Funktion stehen muss. In diesem Codeabschnitt steht der Befehl, genau wie im Skript selbst, auf der letzten Zeile. Mit dem Attribut „new“ wird für die Funktion Speicher reserviert und die Methode „run“ startet das Programm.

Innerhalb der Funktion folgt nun der Befehl „$.writeln“, welcher während der Entwicklungsphase Strings[33] oder Variablen in der Konsole ausgibt, sobald der Programmablauf dort angelangt ist. Das ist sehr hilfreich, wenn während der Produktion im Code Fehler auftreten, die damit eingegrenzt werden können. Danach werden zwei Variablen deklariert. Zuerst wird die aktuelle Funktion, welche unter dem Platzhalter „this“ zu finden ist in die Variable „Rig“ geschrieben. Danach flogt eine zweite Variable, die den Namen „desc“ erhält und einen String mit der Beschreibung des Skriptes aufnimmt. Dieser wird später beim Klicken des About-Buttons abgerufen. Als Letztes fragt das Skript die Versionsnummer von After Effects ab. Ist diese kleiner als sieben wird ein Fehlerdialog ausgeführt, der dem Nutzer mitteilt, dass mit dieser Version das Skript nicht funktioniert. Ist die Version größer oder gleich sieben fährt das Skript mit den nächsten Programmzeilen fort.

33 Zeichenkette, die als eigenständiger Datentyp betrachtet wird

7.2 Sprachfunktion

```
langStr = function (str) //Sprachfunktion
{                   switch (app.language) {

                                case Language.GERMAN : return str.ger;
                                break;

                                case Language.ENGLISH : return str.en;
                                break;

                                case Language.ITALIAN : return str.it;
                                break;

                                case Language.FRENCH : return str.fr;
                                break;

                                case Language.SPANISH : return str.sp;
                                break;

                                };

};
```

Listing 2: Deklaration der Sprachfunktion

Um fast allen After Effects-Nutzern gerecht zu werden wird als Nächstes die Sprachfunktion eingebunden. Dies hat nicht nur den Grund, dass das Skript in der Oberfläche den jeweiligen Sprachen angepasst wird, sondern vielmehr, um die programmiertechnischen Ausdrücke von jeder After Effects-Sprachversion individuell zu ersetzen. Was das bedeutet wird im nächsten Punkt geklärt. Die in Listing 2 eingeführte Funktion beinhaltet lediglich einen Schaltoperator, der die ausgeführte, Sprachversion von After Effects abgleicht und bei einer Übereinstimmung eine Variable der jeweiligen Sprache zurück gibt.

```
var Expr_Pt =       {en:"Point Control", ger:"Einstellungen für Punkte",
                    fr:"Paramètre point d'effet", it:"Controllo punto", sp:"Control de punto"};
var Expr_Wk =       {en:"Angle Control", ger:"Einstellungen für Winkel",
                    fr:"Paramètre angle", it:"Controllo angolo", sp:"Control de ángulo"};
```

Listing 3: Sprachstrings

Listing 3 zeigt was der Hauptgrund für die Mehrsprachigkeit des Skriptes ist. Die Expressionregler, die später in den Expressions genutzt werden, müssen in jeder Sprachversion von After Effects korrekt benannt werden. Wird dies nicht getan, erhält man eine Fehlermeldung und das Skript ist nicht ausführbar. Hier sind auch die einzelnen Sprachvariablen zu sehen, welche in Listing 2 bestimmt wurden. Im späteren Verlauf werden noch oft Variablen an die Sprachfunktion übergeben, welche die richtigen Ausdrücke für die jeweilige Sprache ersetzen.

7.3 Benutzeroberfläche

Abbildung 9: Benutzeroberfläche

After Effects bietet dem Skriptprogrammierer eine Reihe hilfreicher, vorgefertigter Userinterface-Elemente.[34] Wie in Abbildung 9 zu sehen, wurden für dieses Skript Dropdownmenüs, Radiobuttons, Eingabefelder und Klickbuttons verwendet. Im linken Bereich werden die eingeladenen Körperteile angezeigt und bei Bedarf per Dropdownmenü dem korrekten Slot zugeordnet. Dann folgen oben rechts Radiobuttons, welche die Ausrichtung des Charakters festlegen. Darunter wird mit einem Eingabefeld die Größe der Ankerpunktdummys (in Pixeln) bestimmt. Besonders bei großen oder kleinen Charakteren kann diese Funktion hilfreich sein. Danach schließt sich der Block mit den Hauptmenüpunkten an, der die vier Phasen zur Charaktererstellung beinhaltet. Den Schluss bildet der About-Button, durch den man sich Versionsinfos und Urheberrechte anzeigen lassen kann.

```
var res=
          „group {                                                    \
          orientation:'row', spacing:15, margins:0,                   \
          alignChildren:[,right','top'],                              \
          ct2: Panel {                                                \
                    text:'"+langStr(P_bp)+"',                         \
                    preferredSize:[200, 350],                         \
                    alignChildren:[,fill','top'],                     \
                    margins:[10,25,10,10],                            \
                    g1: Group {                                       \
                              orientation:'row', spacing:20,          \
                              x1: StaticText { text:'"+langStr(P_     \
                              head)+"', alignment:[,left','left']},   \
                              ddl1: DropDownList { size:[150,15],     \
                              alignment:[,right','right']},           \
                                        },                            \
                              },                                      \
                    },                                                \"
```

Listing 4: Auszug des Interfacestrings

In Listing 4 gut zu erkennen wird der Code in einen langen String geschrieben, welcher alle Spezifikationen enthält, die für das Userinterface benötigt werden. Besonders fällt die Ähnlichkeit zur HTML-Ergänzungssprache CSS[35] auf, denn auch hier kann man zusammengehörige Programmteile zu Containern, gekennzeichnet mit einer Umrandung, zusammenführen und per Float-Befehl dynamisch aneinander ausrichten. Der komplette String wird dabei in die Variable „res“ geschrieben und setzt sich wie folgt zusammen.

```
          „group {                                                    \
          orientation:'row', spacing:15, margins:0,                   \
          alignChildren:[,right','top'],                              \
```

Listing 5: Einstellungen für das Skriptfenster

Zunächst wird eine globale Gruppe deklariert welche keine sichtbaren Ausgaben bewirkt, aber die Eigenschaften für die darin enthaltenen Panels (Paletten) und Container festlegt. So sollen alle untergeordneten Elemente in einer Reihe, rechts-oben und mit einem Abstand von fünfzehn Pixeln ausgerichtet werden.

```
          ct2: Panel {                                                \
                    text:'"+langStr(P_bp)+"',                         \
                    preferredSize:[200, 350],                         \
                    alignChildren:[,fill','top'],                     \
                    margins:[10,25,10,10],                            \
```

Listing 6: Einstellungen für den linken Container

Als Nächstes folgen die Eigenschaften eines untergeordneten Panels, welches per Textattribut im Skriptfenster einen Namen bekommt. Hier wird zum ersten Mal die Sprachfunktion aufgerufen. Durch das Attribut „P_bp“ weiß die Funktion, dass es sich dabei um ein Panel für „bodyparts“, also Körperteile, handelt. Da die Sprachversion von After Effects zu Beginn schon ermittelt wurde, wird jetzt nur noch der korrekte Wert in dem angezeigten Array ausgewählt und im String ersetzt. Als Nächstes wird die Größe des Containers festgelegt und wie die darin enthaltenen Elemente ausgerichtet wer-

35 Cascading Stylesheets, mit der HTML-Elemente exakt formatiert und positioniert werden können (http://de.selfhtml.org/css/)

den. In diesem Fall gilt für die Unterobjekte sich so aufzuteilen, dass die ganze Breite genutzt und dabei oben begonnen wird. Zum Schluss werden noch die Abstände zueinander festgelegt, dass es auch optisch ansprechend und übersichtlich aussieht.

```
                        orientation:'row', spacing:20,                    \
                        x1: StaticText { text:'"+langStr(P_               \
                        head)+"', alignment:[,left','left']},             \
                        ddl1: DropDownList { size:[150,15],               \
                        alignment:[,right','right']},                     \
                                  },                                      \
                        },                                                \
                },                                                        \"
```

Listing 7: Einstellungen für den Inhalt des linken Containers

Darunter befinden sich, im Falle des Körperteilpanels, die Dropdownmenüs der einzelnen Charakterteile. Da jedes Element aus einem Beschreibungstext und dem Menü selbst besteht, wird wieder mit der Orientierung der Teile begonnen. Da diese nebeneinander stehen sollen bekommt die Ausrichtung das Attribut „row“. Der Abstand zwischen dem Beschreibungstext und dem Menü soll 20 Pixel sein. Als Letztes folgt die Definition der Inhalte. Dabei gibt ein Doppelpunkt an, dass der nachfolgende Ausdruck den Typ des Objektes festlegt. Als Erstes wird hier ein statischer Text erstellt mit einer linken Ausrichtung und einem Namen aus der Sprachenvariablen. Daneben soll das dazugehörige Dropdownmenü erscheinen, also wird als Typ „DropDownList“ mit einer festen Größe und mit der dazugehörigen Ausrichtung ausgewählt.

Die folgenden Dropdownmenüs werden analog erstellt. Man muss nur darauf achten, dass die folgenden Menüs keine Kinder von dem ersten Dropdownmenü sind, sondern auf der gleichen Ebene unter dem Panelcontainer liegen. Als Nächstes würde das Controlpanel mit den darin enthaltenen Containern und Buttons folgen.

7.4 Laden der Körperteile

```
pal.gr.ct3.ct5.g15.getLayBtn.onClick = function () //Ebenen Laden
{

var comp = app.project.activeItem;
          var lists = [
          pal.gr.ct2.g1.ddl1,
          pal.gr.ct2.g2.ddl2,
          pal.gr.ct2.g3.ddl3,
          pal.gr.ct2.g4.ddl4,
          pal.gr.ct2.g5.ddl5,
          pal.gr.ct2.g6.ddl6,
          pal.gr.ct2.g7.ddl7,
          pal.gr.ct2.g8.ddl8,
          pal.gr.ct2.g9.ddl9,
          pal.gr.ct2.g10.ddl10,
          pal.gr.ct2.g11.ddl11,
          pal.gr.ct2.g12.ddl12,
          ];

          for (var i = 1; i <= comp.numLayers; i++)
          {
                    for (var j = 0; j < lists.length; j++)
                    {
                              if (i == 1) lists[j].removeAll();
                              lists[j].add(„item“, comp.layer(i).name);//hinzufuegen Dropdownlisten
                    };
          };
          for (var n = 0; n < lists.length; n++)
          {
                    lists[n].add(„item“, „none“);
                    lists[n].selection = n; //Vorauswahl festlegen
          };
};
```

Listing 8: Dropdownmenüs mit Ebenen

Sobald auf den Button „Ebenen laden“ geklickt wird, ruft das Skript in der ersten Zeile des Listings 8 eine neue Funktion auf. Begonnen wird diese mit der Deklaration der Variable „comp“, welche den Ausdruck „app.project.activeItem“ aufnimmt. Im Einzelnen weist dieser Codeabschnitt auf das aktuell aktiv ausgewählte Element hin. Dabei wird zunächst die Applikation, dann das Projekt und zum Schluss das aktive Objekt angesprochen und damit auch aufgenommen. In diesem Fall ist das die Komposition mit den zwölf darin befindlichen Körperebenen. Als Nächstes wird ein Array mit dem Namen „lists“ erstellt, wobei die einzelnen Dropdownmenüpunkte darin aufgenommen werden. Dieser Schritt wird unternommen, um die Menüpunkte später leichter ansprechen zu können. Das Prinzip ist ähnlich, um an das richtige Objekt zu gelangen, wird auch hier hierarchisch von oben nach unten vorgegangen. Die oberste Instanz ist das Skriptfenster selbst, was mit „pal“ bezeichnet wird. Danach folgt die Gruppe die alle Dropdownmenüs in sich vereint, gefolgt vom Panel „Körperteile“, welchem der Ausdruck „ct2“ zugeordnet ist. Am Ende der Hierarchie trifft man auf die Gruppe, die den dazugehörigen statischen Text und die mit „ddl“ bezeichnete Dropdownliste enthält. Davon gibt es für jedes Charakterteil jeweils einen Eintrag.

Als Nächstes müssen drei Programmschleifen durchlaufen werden, um die Ebenen der ausgewählten Komposition in das Skript zu übertragen. Die äußere Schleife durchläuft alle Ebenen, beginnend bei der 1. und stoppt erst wenn alle einmal angesprochen wurden. Die innere Schleife hangelt sich durch das gesamte Array aus Drop-

downmenüs. Eine zusätzliche If-Anweisung verhindert beim wiederholten Ausführen der Funktion, dass Werte gedoppelt werden. Jedes Mal wenn die erste Ebene angelaufen wird, leert die If-Abfrage den Cache der Dropdownmenüs. Als Nächstes wird mit der Anweisung „add“ gefolgt durch den String „item“ ein neues Element dem Dropdownmenü hinzugefügt. Im Skript ist das die Ebene, die gerade durchlaufen wird. Nun werden alle Dropdownmenüs mit allen zwölf Ebenen bestückt, wonach in der nächsten Schleife eine Vorauswahl getroffen wird. Dies hat zur Folge, dass noch einmal jedes Dropdownmenü, von oben angefangen, eine individuelle Ebene als Standardauswahl zugewiesen bekommt.

7.5 Ankerpunkte setzen

```
pal.gr.ct3.ct5.g16.anchorPBtn.onClick = function () //Ankerpunkte setzen
{
          app.beginUndoGroup(„make anchor points“);
          if (pal.gr.ct3.ct4.rightBtn.value != true && pal.gr.ct3.ct4.frontBtn.value != true
          && pal.gr.ct3.ct4.leftBtn.value != true)
          {         alert(langStr(A_no_mode), „RigIt“);     };
          else
          {
          var comp = app.project.activeItem;

          var lists2 = [
                    pal.gr.ct2.g1.ddl1.selection,
                    pal.gr.ct2.g2.ddl2.selection,
                    pal.gr.ct2.g3.ddl3.selection,
                    pal.gr.ct2.g4.ddl4.selection,
                    pal.gr.ct2.g5.ddl5.selection,
                    pal.gr.ct2.g6.ddl6.selection,
                    pal.gr.ct2.g7.ddl7.selection,
                    pal.gr.ct2.g8.ddl8.selection,
                    pal.gr.ct2.g9.ddl9.selection,
                    pal.gr.ct2.g10.ddl10.selection,
                    pal.gr.ct2.g11.ddl11.selection,
                    pal.gr.ct2.g12.ddl12.selection,
                    ];
          var Ap_list = [Ap_head, Ap_neck, Ap_ut, Ap_lt, Ap_ua, Ap_la, Ap_ha,
                         Ap_fi, Ap_ul, Ap_ll, Ap_fo, Ap_to];

          for (var i = 1; i <= comp.numLayers; i++)
          {
                    comp.layer(i).locked = true;
          };
```

Listing 9: Vorbereitung zum Erstellen der Ankerpunktdummys

Wenn jedes Körperteil in dem dazugehörigen Dropdownmenü ausgewählt wurde können nun die Ankerpunktobjekte generiert werden. Im Listing 9 wird wieder mit einer Funktion begonnen, welche ausgeführt wird sobald der Button zum Setzen der Ankerpunkte gedrückt wurde.

In der darauffolgenden Zeile initialisiert das Skript eine „Undo“-Gruppe, was dafür sorgt, dass alle Befehle innerhalb dieser Gruppe mit dem programminternen Befehl „rückgängig“ widerrufen werden können. Damit die richtigen Platzierungszonen für die Gelenke, in After Effects auch Ankerpunkte genannt, angezeigt werden können, muss zunächst abgefragt werden, welche Ausrichtung der Nutzer für seinen Charakter gewählt hat. Falls kein Radiobutton ausgewählt wurde, gibt das Skript eine Fehlermeldung aus. In

die Variable „lists2“ lädt nun das Skript nacheinander aus den Dropdownmenüs die jeweils aktuell ausgewählte Ebene. Diese wird später benötigt, um die Ankerpunktdummys ihren richtigen Positionen zuzuweisen. Dann speichert ein neues Array-Objekt alle Gelenkbezeichnungen, welche zu Beginn des Skriptes in den fünf Sprachen festgelegt worden sind. Das Ende dieser Abbildung bildet eine Schleife, welche alle Ebenen schließt, also für den Benutzer unveränderbar macht, und somit keine Körperteile mehr verschoben werden können. Damit wird Fehlern des Nutzers vorgebeugt, die beim späteren Zusammensetzen des Charakters Probleme verursachen würden.

```
var ap_dm = parseFloat(pal.gr.ct3.g13.ET.text);

//Deckkraft des Ankerpunktdummys
var op = 25;

//Erstellung des Ankerpunktdummys
var Ap_head = comp.layers.addSolid([1.0,0.0,0.0], langStr(Ap_he), ap_dm, ap_dm, 1);
Ap_head.position.setValue(comp.layer(pal.gr.ct2.g1.ddl1.selection).position.value);
var Ap_head_mask = Ap_head.Masks.addProperty(„Mask“);
Ap_head_mask.inverted = false;
var Ap_head_maskShape = Ap_head_mask.property(„maskShape“);
var Ap_head_Shape = Ap_head_maskShape.value;
Ap_head_Shape.vertices = [[(ap_dm/2),0],[ap_dm,(ap_dm/2)],[(ap_dm/2),ap_dm],[0,(ap_dm/2)]];
Ap_head_Shape.inTangents = [[-(ap_dm/4),0],[0,-(ap_dm/4)],[(ap_dm/4),0],[0,(ap_dm/4)]];
Ap_head_Shape.outTangents = [[(ap_dm/4),0],[0,(ap_dm/4)],[-(ap_dm/4),0],[0,-(ap_dm/4)]];
Ap_head_Shape.closed = true;
Ap_head_maskShape.setValue(Ap_head_Shape);
Ap_head.scale.expression = „[100,100];“;
```

Listing 10: Erstellung eines Ankerpunktdummys

Im nächsten Schritt wird als Erstes das Dummytextfeld ausgelesen, um die vom Nutzer gewünschte Größe an die Gelenkpunkte zu übertragen. Als Standardwert werden 10 Pixel im Durchmesser vorgegeben. Die nächste Zeile führt die Variable „op“, die als Abkürzung für opacity, also Deckkraft, steht ein. Als Wert erhält diese 25, was 25% Deckkraft entspricht. Der nun folgende Textblock generiert an der aktuellen Position eines Körperteils eine rote Farbfläche, die zur besseren Darstellung mit einer runden Maske ausgestattet wird.

Als Beispiel ist hier die Erstellung des Kopfgelenks dokumentiert. Begonnen wird mit einer neuen Farbfläche, welche durch die Parameter ([1.0, 0.0, 0.0]), die Farbe rot zugewiesen bekommt. Dabei erstrecken sich die einzelnen Parameter von null bis eins und sind in Rot, Grün und Blau aufgeteilt. Da nur der Rotanteil den maximal Wert erhält, ist die Farbfläche automatisch rot gefärbt. Als nächste Eigenschaft wird der Name festgelegt, welcher mit dem, aus Punkt 7.2 bekannten, Sprachenstring ersetzt wird. Danach erhält die Farbfläche die Breiten- und Längenangaben, die vom Dummygrößentextfeld übernommen werden. Den Schluss bildet die zeitliche Länge, welche die Farbfläche in der Komposition einnimmt, wobei der Wert eins einer Sekunde entspricht.

Die Farbfläche wurde nun generiert, hat aber noch keine Position innerhalb der Komposition. Diese erhält sie direkt von der ausgewählten Ebene des Dropdownmenüs in der nächsten Zeile. Um dem Erscheinungsbild eines Punktes nachzukommen, wird nun eine kreisförmige Maske hinzugefügt. Diese wird als Erstes invertiert, da der

äußere Teil nicht sichtbar sein soll. Um das Aussehen der Maske festzulegen wird nun die Maskenform mit den dazugehörigen Punkten, die in After Effects auch Vertices genannt werden, ausgestattet. Diese werden jeweils in die Mitte der vier Seiten des Quadrates gesetzt. Da die Punkte bisher noch mit geraden Linien verbunden sind, aber ein Kreis entstehen soll, müssen nun Tangenten entlang der Seiten des Quadrates gezogen werden, bis der Pfad so weit gebogen ist, dass ein Kreis entsteht.

In den letzten drei Zeilen wird der Maskenpfad geschlossen, die Form der Maske zugeordnet und die Skalierung des Ankerpunktdummys auf 100% in x und y begrenzt. Der gesamte Ablauf wird nun so lang fortgesetzt bis alle anderen Körperteile mit einem Gelenkpunkt versehen wurden. Bevor diese Phase abgeschlossen ist, müssen noch die Platzierungszonen durch weitere Farbflächen generiert werden. Diese geben dem Nutzer vor, in welchen Bereichen Ankerpunkte gesetzt werden können. Dadurch beugt man Fehlern beim Zusammensetzen des Charakters vor.

7.6 Aufbauen des Charakters

In diesem Arbeitsschritt wird der Charakter anhand der gegebenen Daten zusammengesetzt. Zu Beginn müssen diverse Voreinstellungen getroffen werden, damit der Hauptteil reibungslos funktioniert.

```
pal.gr.ct3.ct5.g17.exeBtn.onClick = function () //Aufbauen des Charakters
          {
          app.beginUndoGroup(„Building Guy“);
          var comp = app.project.activeItem;
          var w = comp.width;
          var h = comp.height;

          for (var i = 1; i <= comp.numLayers; i++)
          {
                    comp.layer(i).locked = false;
          };
          for (var j = 1; j <= 12; j++)
          {
                    comp.layer(1).remove();
          };
```

Listing 11: Vorbereitung der Aufbauphase

Der Codeabschnitt in Listing 11 beginnt wieder mit einer Funktion, die beim Drücken des „Charakter aufbauen“-Knopfes ausgeführt wird. Um auch diesen Schritt einfach in After Effects rückgängig zu machen, wird als Erstes eine Undo-Gruppe erstellt. Danach deklariert das Skript die Variable „comp“ und direkt danach die Variable „w“ und „h“ welche die Kompositionsbreite und -Höhe beinhalten. Damit die Körperteile bewegt werden können, werden sie mit einer ersten Schleife wieder freigesetzt. Die zweite Schleife löscht die Farbflächen, welche die Platzierungszonen der Gelenke darstellten.

```
//Laden der Koerperteile aus den Dropdownmenues
var kopf = comp.layer(pal.gr.ct2.g1.ddl1.selection);
[...]
//Namen fuer Koerperteile
kopf.name = langStr(Bp_he);
[...]
//hinzufuegen von Animatorobjekten
var null_l_bein_con = comp.layers.addNull();
          null_l_bein_con.enabled = true;
          null_l_bein_con.name = langStr(C_l_leg);
          null_l_bein_con.moveToBeginning();
[...]
//neuanordnung der Teile innerhalb der Zeitleiste
kopf.moveToEnd();
[...]
```

Listing 12: Benennung der Körperebenen und hinzufügen von Kontrollobjekten

In Listing 12 werden die noch fehlenden Teile des Rigs hinzugefügt. Als Erstes speichert das Skript die Dropdownmenüauswahlen in die jeweiligen Variablen. Als Beispiel dient wieder der Kopf des Charakters. Danach werden die Ebenen in der Zeitleiste durch die schon bekannte Sprachfunktion umbenannt. Nachdem das Skript alle zwölf Teile umbenannt hat, werden als Nächstes die ersten Nullobjekte erstellt. In After Effects dienen Nullobjekte als Hilfsobjekte, die alle Transformationen ausführen können, aber in der finalen Ausgabe nicht gerendert werden. Als Erstes muss eine Variable erstellt werden, welche das neue Nullobjekt aufnimmt. Nun definiert man die Sichtbarkeit des Nullobjekts mit „wahr“, gibt ihm einen Namen und verschiebt es in der Zeitleiste an die erste Stelle. Diese Prozedur wird nun mit allen restlichen Kontrollobjekten durchlaufen. Im darauffolgenden Schritt müssen auch die Körperebenen neu angeordnet werden, um anatomisch unmögliche Überlappungen zu vermeiden.

```
//duplicating parts
var R_bein_01 = L_bein_01.duplicate();
          R_bein_01.name = langStr(Bp_ul_r);
          R_bein_01.moveAfter(torso_02);
[...]
//adding dummynulls
var null_l_schulter_dum = comp.layers.addNull();
          null_l_schulter_dum.enabled = false;
          null_l_schulter_dum.name = langStr(D_l_shoulder);
          null_l_schulter_dum.moveAfter(hals_01);
```

Listing 13: Duplizieren fehlender Körperteile und hinzufügen von Hilfsobjekten

Da die zu animierende Figur immer noch ohne zweiten Arm und zweites Bein ausgekommen ist, wird im nächsten Schritt das bestehende Teil dupliziert. Als Beispiel sieht man in Listing 13 den linken Oberschenkel, welcher mit dem Befehl „duplicate“ gedoppelt, und direkt in die Variable „R_bein_01“ geschrieben wird. Nun muss nur noch ein neuer Name und eine neue Position in der Zeitleiste an das neue Körperteil übertragen werden. Als letzten Schritt werden nochmals Nullobjekte erstellt, mit dem Unterschied, dass diese für den Nutzer nicht sichtbar sind und damit nur als Hilfsobjekte dienen. Sie werden später dazu dienen, Justierungen nach dem Verknüpfen des Charakters zu vereinfachen und Expressions, welche für das Animieren des Charakters notwendig sind, aufzunehmen.

```
//Uebertragung von Ankerpunktpositionen an Koerperteile
var head_ap = kopf.position.value - comp.layer(langStr(Ap_he)).position.value;
var head_ap2 = kopf.anchorPoint.value - head_ap;
kopf.anchorPoint.setValue(head_ap2);
[...]
```

Listing 14: Übertragung der Ankerpunkte an die Charakterebenen

Alle Vorbereitungen sind erledigt und, wie in Listing 14 zu sehen, überträgt das Skript die Position der Ankerpunktdummys an die einzelnen Körperteile.

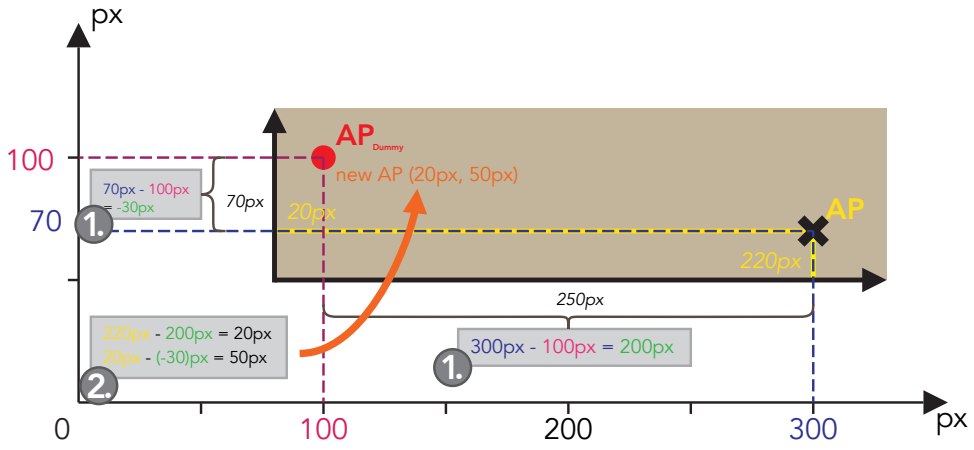


Abbildung 10: Schema zur Übertragung von Gelenken

Abbildung 10 gibt Hilfestellung bei der Erklärung. Wie im Codebeispiel am Anfang zu sehen ist, wird zunächst die Position des Ankerpunktdummys von der Position des aktuellen Körperteils subtrahiert. Diese Rechnung ist notwendig, da Ankerpunktpositionen nicht im globalen Koordinatensystem angegeben werden, sondern von der Breite und Höhe des Körpers abhängen. Der Nullpunkt befindet sich dabei immer in der linken unteren Ecke der Ebene. In Abbildung 10 erklärt Punkt 1 den Vorgang. Aus der Berechnung ergeben sich die Abstände in x-und y-Richtung der beiden Elemente. Um den Ankerpunkt richtig zu setzen muss jetzt vom aktuellen Ankerpunkt das Ergebnis aus Schritt eins abgezogen werden, was auch im 2. Punkt in Abbildung 10 nachvollziehbar dargestellt ist. In der letzten Phase wird nur noch das Ergebnis als neuer Ankerpunkt gespeichert.

Wenn alle Körperteile den Prozess durchlaufen haben startet die Aufbauprozedur. Wichtig bei diesem Schritt ist vor allem, dass der Algorithmus bei möglichst vielen unterschiedlichen Charakteren akzeptable Ereignisse erzielt. Das Prinzip ist recht einfach und wird anhand des Listings 15 und einer Grafik erklärt.

```
//Aufbau des Charakters
if(pal.gr.ct3.ct4.leftBtn.value == true)
{
          var w = comp.width;
          var h = comp.height;
          var ori = [w/2, (h/10*9)]
//Position fuer Zehen setzen
          R_zeh.position.setValue(ori);
          L_zeh.position.setValue(ori);

          L_fuss.anchorPoint.setValue([L_fuss.width, L_fuss.height]);
          R_fuss.anchorPoint.setValue([R_fuss.width, R_fuss.height]);

//Position fuer rechten Fuss setzen
var pos_R_fuss = [R_zeh.position.value[0]+R_fuss.width, R_zeh.position.value[1] + R_zeh.height - R_
zeh.anchorPoint.value[1]+R_fuss.anchorPoint.value[1]-R_fuss.height];
R_fuss.position.setValue(pos_R_fuss);

//Position fuer linken Fuss setzen
var pos_L_fuss = [L_zeh.position.value[0]+L_fuss.width, L_zeh.position.value[1] + L_zeh.height - L_
zeh.anchorPoint.value[1]+L_fuss.anchorPoint.value[1]-L_fuss.height];
L_fuss.position.setValue(pos_L_fuss);
```

Listing 15: Aufbau des Charakters

Als Erstes muss geprüft werden welche Ausrichtung der Charakter durch den Nutzer vorgegeben bekommt. Im Codebeispiel wird angenommen, dass der User den Button für eine linke Ausrichtung gedrückt hat. Darauffolgend werden drei Variablen deklariert. Das Skript speichert in der Variable „w“ die Kompositionsbreite, in „h“ die Kompositionshöhe und in „ori“ den Ursprung der Charaktererstellung. Dieser liegt immer auf der Hälfte der Kompositionslänge und auf 9/10 der Kompositionshöhe. Beim Aufbau von unten nach oben können nun beide Zehen als Erstes Körperteil am Ursprung platziert werden. Die beiden Zehen bekommen per „setValue“-Befehl die Positionswerte der Variable „ori“ zugewiesen. Durch das Setzen der Fußankerpunkte auf die Fersen der Füße, werden die ersten Vorbereitungen für den korrekten Ablauf des Abrollmechanismus getroffen. Um die Fußankerpunkte dort zu platzieren, wird jeweils die Breite des Fußes als x-Wert und die Höhe als y-Wert des Ankerpunktes gesetzt. Nun muss der Fuß selbst noch die richtige Position einnehmen. Eine neue Variable nimmt die Positionswerte auf, welche direkt in ein Array, also eine Variable mit zwei Werten geschrieben wird. Der erste Wert legt den x-Wert der neuen Fußposition fest. Wie man auch in Abbildung 11 sehen kann, wird die Ausgangsposition des schon richtig gesetzen Zehs hergenommen und darauf die Länge des Fußes hinzuaddiert. Das darauffolgende Komma trennt den x-Wert vom y-Wert, welcher mit der y-Position des Zehs bestückt wird. Damit auch die zwei Körperteile in einer Ebene auf dem Boden stehen, selbst bei unterschiedlicher Höhe, wird dies nun mit den Ankerpunktwerten und den jeweiligen Höhenwerten der Ebenen ausgeglichen. Das Gleiche wird für den rechten Fuß umgesetzt und danach für alle folgenden Elemente. Es müssen also nie feste Positionswerte hergenommen werden, um den kompletten Charakter vom Zeh bis zum Kopf zusammenzusetzen.

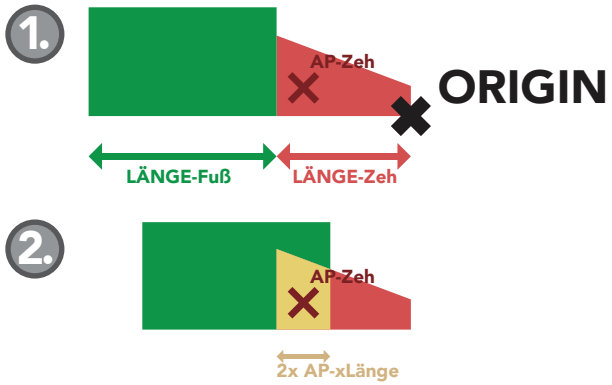


Abbildung 11: Positionierung anhand der Gelenkplatzierung

Die einzigen festen Werte werden von der Kompositionsgröße in After Effects vorgegeben. Daraus resultiert der Ursprung und die Position der Zehen, welche auch die Grundlage für den Rest des Körpers sind. Zusätzlich zu den Körperteilen müssen auch alle Helferelemente an die richtige Position gesetzt werden. Wenn alle Schritte abgeschlossen sind und alle Ankerpunkte übertragen wurden, löscht das Skript zum Schluss die Dummygelenke. Aufgrund der verschiedenen Ausrichtungen müssen gewisse Elemente unterschiedlich angeordnet werden. So erhalten die Zehen bei der Frontansicht nicht die gleiche Position, sondern unterscheiden sich im x-Wert, der sich an der Breite des Torsos orientiert. Bei einem solchen Algorithmus ist es fast nicht auszuschließen, dass Änderungen an der Positionierung im Nachhinein vorgenommen werden müssen. Im Gegenteil, dies ist sogar gewünscht, denn so erhält der Nutzer den Freiraum für weitere Experimente.

7.7 Verlinkung des Charakters

In diesem Abschnitt sieht der Charakter aus, wie er später auch animiert werden soll. Auch wenn nach dem Aufbauteil weiterhin fehlplazierte Elemente vorhanden sind, können diese dennoch im fertigen Charakter nachträglich verändert werden. Wie bei allen anderen Phasen wird auch hier die Variable „comp“ initialisiert und die Namen der Körperteile vergeben. Sind alle Elemente bis hin zu den Hilfsobjekten benannt, werden im nächsten Schritt die Abhängigkeiten zwischen den Körperteilen erzeugt.

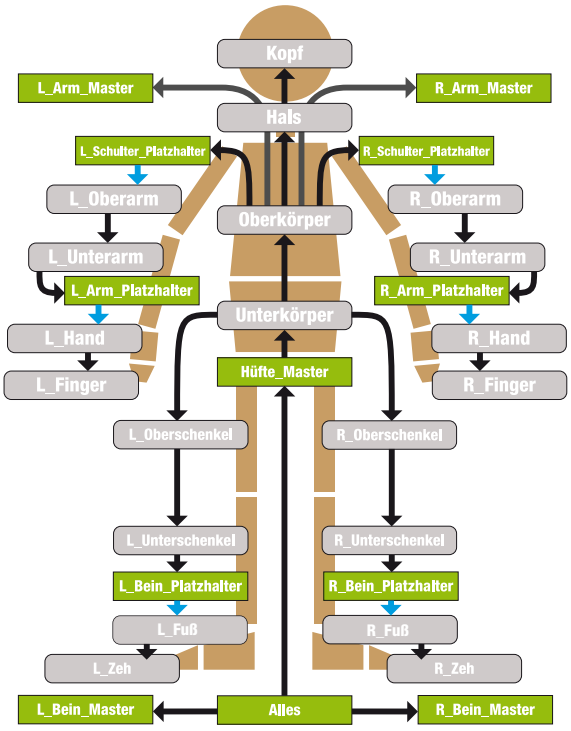


Abbildung 12: Hierarchiestruktur

Das Resultat des gesamten Prozesses stellt eine hierarchische Struktur wie in Abbildung 12 dar, die durch eine Eltern-Kind-Beziehung erreicht wird. Durch den „parent“-Befehl wird den Elementen das jeweils übergeordnete Objekt zugeteilt. Für eine Ebene kann es dabei immer nur ein Elternteil geben, wobei ein Elternteil aber viele Kinder haben kann. Außerdem können auch Positionswerte per Expression an andere Ebenen übertragen werden, diese Verbindung ist in Abbildung 12 mit blauen Pfeilen dargestellt. Dabei sind die grauen Objekte die aktiven Körperteile und die grünen stellen die Hilfs- und Steuerobjekte dar.

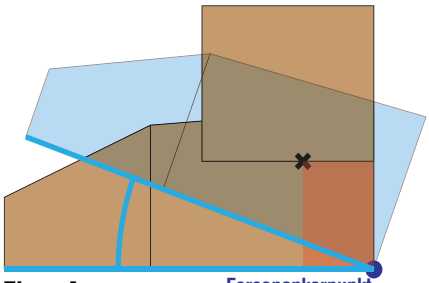
Jetzt fügt das Skript den Ebenen, welche später die Expressions steuern, Effekte hinzu. Die im Skript erstellten Effekte wirken sich dabei nicht auf das generierte Bild aus, sondern geben dem Nutzer Kontrollmechanismen für eine komfortablere Animation des Charakters.

```
var Rot_Kopf = null_master_con.property(„Effects“).addProperty(langStr(Expr_Wk));
                              Rot_Kopf.name = langStr(E_rot_head);
var Rot_Torso = null_master_con.property(„Effects“).addProperty(langStr(Expr_Wk));
                              Rot_Torso.name = langStr(E_rot_torso);
var Rot_Hals = null_master_con.property(„Effects“).addProperty(langStr(Expr_Wk));
                              Rot_Hals.name = langStr(E_rot_neck);
var Trenner_01 = null_master_con.property(„Effects“).addProperty(langStr(Expr_Sl));
                              Trenner_01.name = „================“;
var L_Roll = null_master_con.property(„Effects“).addProperty(langStr(Expr_Wk));
                              L_Roll.name = langStr(E_l_roll);
var L_Toe = null_master_con.property(„Effects“).addProperty(langStr(Expr_Wk));
                              L_Toe.name = langStr(E_l_toe);
var L_Roll_Display = null_helper_con.property(„Effects“).addProperty(langStr(Expr_Pt));
                              L_Roll_Display.name = langStr(E_l_roll_display);
var L_Heel_Display = null_helper_con.property(„Effects“).addProperty(langStr(Expr_Pt));
                              L_Heel_Display.name = langStr(E_l_heel_display);
var L_Toe_Display = null_helper_con.property(„Effects“).addProperty(langStr(Expr_Pt));
                              L_Toe_Display.name = langStr(E_l_toe_display);
```

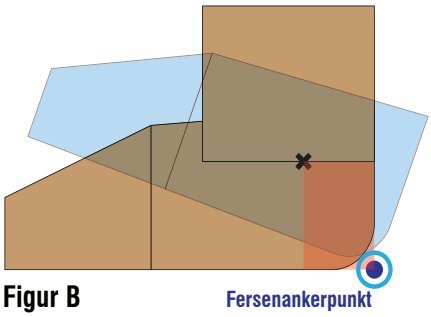
Listing 16: Hinzufügen von Effekten

In Listing 16 ist gut erkennbar, wie einfach es ist Effekte einer Ebene hinzuzufügen. Zunächst muss dem Effekt ein Variablennamen zugeordnet werden. Im Beispiel ist das „Rot_Kopf“, welcher im fertigen Setup die Rotation des Kopfes steuert. Dann folgen Attribute, welche durch einen Punkt voneinander getrennt sind und nacheinander erläutert werden. Die Ebene „null_master_con“ (con für control) erhält durch den Punkt getrennt, das Attribut „property(„Effects“)“, was für die Ebeneneigenschaft „Effekt“ steht. Durch das nächste Attribut „addProperty“ wird ein Element aus der Eigenschaft Effekte, dessen Name immer in Klammern hinter dem Attribut steht, der Ebene hinzugefügt. Da die Effekte in jeder Sprache auch andere Namen haben, kommt wieder die Sprachfunktion zum Einsatz. Der hinzugefügte Effekt stellt eine visuelle Benutzeroberfläche für Winkeleinstellungen dar, die zur Drehung des Kopfes eingesetzt wird. In der nächsten Zeile erhält der Winkelregler einen Namen, der in der After Effekts Oberfläche sichtbar ist. So werden auch die restlichen Effekte hinzugefügt und im Abschnitt Expressions die Rotationswerte des Reglers mit denen des Kopfes verbunden.

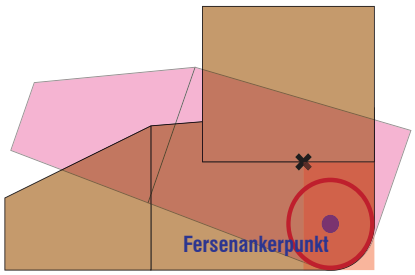
7.8 Fersenankerpunkt



Figur A



Figur B



Figur C

Abbildung 13: Fersenankerpunkt

Der Fersenankerpunkt ist neben dem Ballenankerpunkt für die Funktionsweise der Füße verantwortlich. Besonders beim Auftreten des Fußes während einer Lauf- oder Gehanimation spielt dieser Punkt eine wichtige Rolle. Zunächst wird beschrieben wie das Skript den Punkt erstellt. Es werden die gleichen Funktionen benutzt, die schon bei der Darstellung der Gelenke zum Einsatz kamen, bis auf den Unterschied, dass der Fersenankerpunkt blau ist. Die blaue Farbe verdeutlicht, dass es sich nicht um ein reguläres Gelenk handelt. Außerdem weist auch wieder eine Platzierungszone auf den Bereich der für die Positionierung sinnvoll ist.

Durch die Neuplatzierung des Punktes wird der Rotationspunkt der Ferse neu vergeben.

In Abbilung 13, Figur A sieht man die Ausgangseinstellungen für den Fersenankerpunkt. Der Punkt wird standardmäßig im Ursprung des Fußes, also in dessen Ferse, erstellt, wobei dies meist auch die Einstellung darstellt, welche für eckige Formen am sinnvollsten ist. Anhand des blauen Schattens in Figur A wird die Position simuliert bei der eine Rotation um den Fersenankerpunkt erfolgt.

Figur B veranschaulicht was passiert, wenn dieser Punkt auch bei abgerundeten Formen an der gleichen Stelle verweilt. Wenn bei runden Formen auch an der Position des Fersenpunktes festgehalten wird, hat der Fuß bei der Rotation keinen Kontakt mit der Bodenplatte. Bei umso größeren Rundungen wird diese Unsauberkeit noch deutlicher.

Figur C zeigt die Lösung des Problems. Wird der Punkt innerhalb der roten Fläche platziert ändert sich automatisch der Rotationsursprung und der Fuß hat Kontakt mit dem Boden.

7.9 Expressions

Im aktuellen Zustand des Rigs ist der Charakter so zusammengesetzt, mit Effekten bestückt und untereinander verlinkt, dass als Letztes nur noch die Mechaniken fehlen um eine bequeme Animation zu gewährleisten. Deshalb werden nun diverse Expressions an das Charakterrig übertragen, damit sich der Charakter zum Schluss so verhält wie er es soll.

7.9.1 Allgemeine Einbindung von Expressions

```
Ebene.effect(Effektname))(Attributname).expression = „...“
```

Listing 17: Hinzufügen von Expressions

Das Listing 17 dient als Beispiel wie Expressions einer Eigenschaft hinzugefügt werden. Zunächst muss die jeweilige Ebene in After Effects angesprochen werden, welche im obigen Beispiel einfach nur „Ebene“ heißt. Als Nächstes folgt, mit einem Punkt getrennt, die Eigenschaft worauf die Expression wirken soll. Im Beispiel ist das der Effekt mit dem Namen „Effektname“, der im Skript wieder durch die Sprachfunktion ersetzt wird. Da manche Effekte mehrere Attribute besitzen die animierbar sind, folgt danach in eckigen Klammern der Eigenschaftsname des Effektes. Zum Schluss wird durch den Ausdruck „expression“, welcher wieder durch einen Punkt vom Rest abgetrennt wird, gekennzeichnet, dass jetzt die einzufügende Expression folgt. Da in den Expressions Sprachvariablen und Satzzeichen, wie Zeilenumbrüche als Befehl geschrieben werden, sehen diese im Skript selbst für den normalen Nutzer nicht übersichtlich aus. Aus diesem Grund wird in den folgenden Abschnitten der Code dokumentiert, welcher zum Schluss in den Effekteigenschaften in After Effects zu sehen ist.

7.9.2 Kopf- und Torsorotation

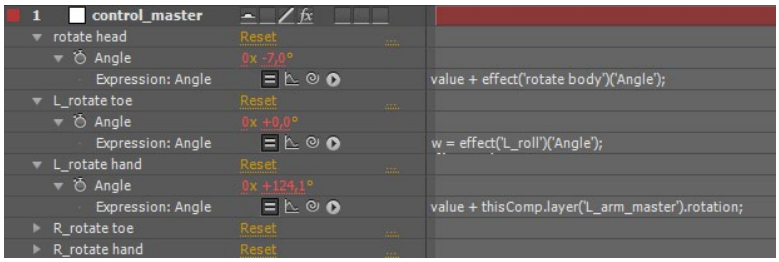


Abbildung 14: Kopf- und Torsorotationsexpression in After Effects

Die Expressions welche den Kopf, Hals und Torso steuern sind vergleichsweise einfach. In Abbildung 14 ist dargestellt wie die Expressions im fertigen Projekt aussehen. Von oben beginnend stellt das weiße Kästchen mit der vorangestellten Eins die erste Ebene in dieser Komposition dar. Darunter sind die Effekte aufgeklappt, die auf dieser

Ebene liegen. In der Abbildung sind zunächst nur die Effekte zu sehen, welche eine Expression erhalten haben.

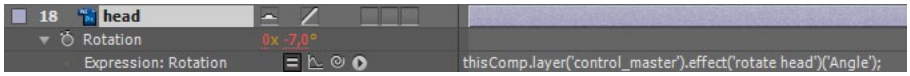


Abbildung 15: Rotationsexpression des Kopfes in After Effects

Damit der Kopf durch einen Kontrollregler, wie in Abbildung 15 gezeigt, bewegt werden kann, wird eine Expression erarbeitet, die den Regler mit dem Rotationsparameter des Kopfes verbindet.

```
thisComp.layer(‚control_master').effect(‚rotate head')(‚Angle');
```

Listing 18: Rotationsexpression des Kopfes

Da diese Expression in der Eigenschaft Rotation eingebettet ist, werden alle Berechnungen, die darin ablaufen auf die Rotation übertragen. Die Zeile in Listing 18 bedeutet rückwärts: Nimm den Wert aus der Eigenschaft „Angle“(Winkel), im Effekt „rotate head“ (Rotationsregler des Kopfes), welcher in der offenen Komposition, in der Ebene „control_master“ zu finden ist. Das heißt es werden alle Änderungen, die in dem Rotationsregler vorgenommen werden auf die Rotation des Kopfes übertragen.

Nun sieht man, dass auch auf der „control_master“-Ebene im Effekt „rotate_head“ wiederum eine Expression enthalten ist. Diese sagt aus, dass der Rotationsregler aus dem „rotate_body“-Effekt, welcher die Rotation des Körpers steuert, auf den Ausdruck „value“ addiert wird. Die Variable „value“ ist eine vorgegebene Variable, die das Überschreiben der ursprünglichen Werte durch Expressions umgeht. Sie gibt immer den aktuellen Wert des Reglers an die Expression weiter, womit wieder Berechnungen erfolgen können. Zu beachten ist hier, dass keine Angaben zur Komposition und der Ebene benötigt werden, da der referenzierte Effekt auf der aktuellen Ebene liegt, auf der auch diese Expression angewendet wird. Durch die Vernetzung unter den jeweiligen Werten werden komfortable Animationsregler für den Nutzer geschaffen.

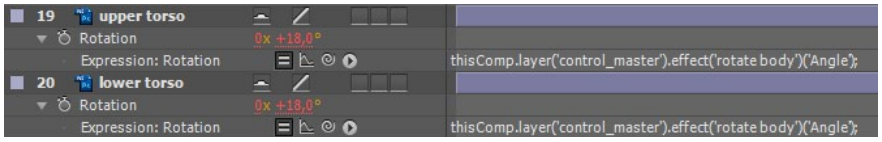


Abbildung 16: Expressions der Torsorotationen in After Effects

```
value + effect(‚rotate body')(‚Angle');
```

Listing 19: Expression der Torsorotation

Auch die beiden Torsoteile werden durch einen Winkelregler gesteuert. Dieser ist auch in diesem Fall der Körperrotationsregler. Analog dazu verhält es sich mit den Rotationen des Halses, der Hände und der Finger, welche sich nur dahingehend unterscheiden, dass sie nicht vom selben Effekt gesteuert werden.

7.9.3 Inverse Kinematik Expression

In diesem Abschnitt wird die Expression für die Inverse Kinematik abgehandelt. Da diese Expression mit trigonometrischen Berechnungen gelöst wird, darf die IK-Kette nur aus zwei Schenkeln und dem einschließenden Gelenk bestehen, sodass immer ein Dreieck und dessen Winkel berechnet werden können. Das ist möglich, weil immer alle drei Punkte, die das Dreieck bilden, nämlich das Oberschenkelgelenk, das Unterschenkelgelenk und die Position des Dummyobjektes ausgelesen werden können. Somit ist die Expression in der Lange die Geraden zwischen den Punkten und die daraus resultierenden Winkel zu berechnen, welche für die Funktion der IK-Kette benötigt werden. Anhand von der Expression für die Bein-IK soll dies in den nächsten Abschnitten erklärt werden.

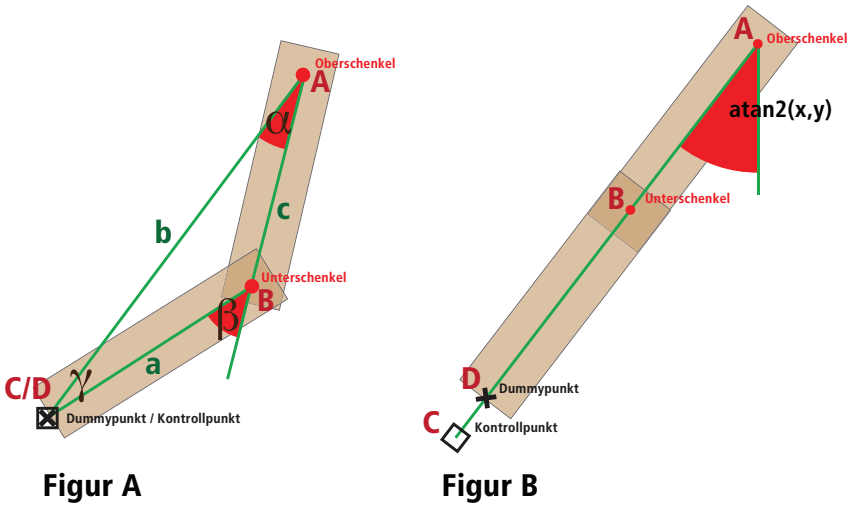


Abbildung 17: IK-Schema

Abbildung 17 stellt die beiden Zustände dar, welche die Objekte unter Einfluss der Inversen Kinematik annehmen können. Die beschriebenen Zustände unterscheiden sich in den Status „eingeknickt“, der in Figur A zu sehen ist und in den Status „ausgestreckt“, welcher durch Figur B dargestellt wird.

Da die IK-Kette auch auf der Eltern-Kind-Methode basiert und dadurch alle Transformationen der Elternteile auf die Kinder übertragen werden, reicht es aus eine Expression nur für die Rotationswerte zu erstellen.

```
upper =             thisComp.layer(,L_upper leg‘);
lower =             thisComp.layer(,L_lower leg‘);
control =           thisComp.layer(,L_leg_master‘);
dummy =             thisComp.layer(,L_leg_dummy‘);
flip =              thisComp.layer(,L_leg_master‘).effect(,ik flip‘)(,Checkbox‘).value;
koerperrotation =   thisComp.layer(,control_master‘).effect(,rotate body‘)(,Angle‘)
hueftenrotation =   thisComp.layer(,hip_master‘).transform.rotation;

A = upper.toWorld(upper.transform.anchorPoint);
B = lower.toWorld(lower.transform.anchorPoint);
C = control.toWorld(control.transform.anchorPoint);
D = dummy.toWorld(dummy.transform.anchorPoint);

laenge =            length(A,B) + length(B,D));
laengeX =           length(A, C);

dx = C[0]-A[0];
dy = C[1]-A[1];
AlphaR =            Math.atan2(dy, dx) - Math.PI/2;
AlphaDeg =          radiansToDegrees(AlphaR);
```

Listing 20: IK-Vorbereitung

Die Expression beginnt in Listing 20 mit der Deklaration der benötigten Variablen und Werte. Diese referenziert die Expression wieder aus anderen Ebenen. Zuerst werden die beiden Schenkel in die Variablen „upper“ und „lower“ gespeichert. Diese verweisen in diesem Beispiel auf den Ober- und Unterschenkel. Danach folgt in der Variable „control“ das Nullobjekt mit dem später das Bein, oder der Arm bewegt werden kann. Als Nächstes lädt die Expression das Hilfsobjekt „dummy“, dass zur Berechnung der Mechanik dient und die Variable „flip“, die für die Ausrichtung der IK-Kette benötigt wird. Die letzten zwei Variablen speichern jeweils die Rotationswerte des Körperreglers und der Hüfte.

Für eine einfachere Berechnung des Dreiecks werden nun die absoluten Positionen in die dafür vorgesehenen Variablen gespeichert. Diese sind wie in Abbildung 17 mit den Buchstaben A bis D benannt. Durch die hierarchische Struktur des Charakterrigs befinden sich die jeweils untergeordneten Objekte in relativer Position zu den übergeordneten. Durch die unterschiedlichen Nullpunkte der Objekte werden Berechnungen erschwert, weshalb der Befehl „toWorld()“ für den Zugriff auf die Weltkoordinaten genutzt wird.

Um unterscheiden zu können, ob das Bein den Status „eingeknickt“ oder den Status „gestreckt“ eingenommen hat, werden nun zwei vergleichbare Variablen eingeführt. Die Variable „laenge“ beinhaltet die zusammenaddierte Länge von Oberschenkel und Unterschenkel, wobei in der zweiten Variable die direkte Strecke vom Ankerpunkt des Oberschenkels bis hin zum Kontrollobjekt enthalten ist.

Als Nächstes wird die Basis für die restlichen Winkel geschaffen. Die Arkustangensfunktion mit zwei Parametern, x und y ermöglicht eine Ausrichtung auf das Kontrollobjekt. Durch das Subtrahieren der jeweiligen x- und y-Werte des Punktes A und des Kontrollobjektes liegt der Ursprung der Arkustangensfunktion im Ankerpunkt des Oberschenkels. Dadurch nimmt die Rotation immer solche Werte an, dass das Bein entlang der Strecke „b“ ausgerichtet bleibt. Im Moment zeigt das Bein jedoch in die

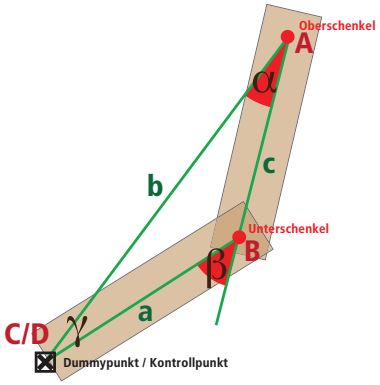
falsche Richtung, weshalb nun „PI/2“ abgezogen werden, um eine Drehung um 180° zu erreichen. Es ist zu diesem Zeitpunkt noch nicht möglich, einfach 180° abzuziehen, da der errechnete Winkel in Radianten angegeben wird. Erst in der letzten Zeile werden durch eine vordefinierte Funktion aus Radianten Gradwinkel.

```
if ( laengeX < laenge )
    {
        a =         length(B,D);
        b =         length(A,C);
        c =         length(A,B);

        AlphaX =    Math.acos( clamp( ( b * b + c * c - a * a ) / ( 2 * b * c ) , -1 , 1 ) );

        if (flip == 1){Alpha = AlphaX * -1 + AlphaR;}
        else {Alpha = AlphaX + AlphaR;}

        x = radiansToDegrees(Alpha) - koerperrotation - hueftenrotation;
    }
else
    {   x = AlphaDeg - koerperrotation - hueftenrotation;}
```

Listing 21: Alpha-Winkel-Expression

Hier in Listing 21 wird der Winkel Alpha berechnet und das Codebeispiel aus Listing 20 weitergeführt. Die weiter oben eingeführten Längenvariablen werden im ersten Schritt mit einer If-Anweisung verglichen, woraus sich die beiden Zustände aus Abbildung 17 ergeben. Ist die Länge zwischen Oberschenkelgelenk und Kontrollobjekt kleiner als die zusammenaddierte Länge der beiden Schenkel, so ist die Bedingung wahr und springt in den darunter befindlichen Codeblock. Dieser wird durch geschweifte Klammern abgegrenzt. Ist die Bedingung falsch, so wird die Else-Anweisung ausgeführt, welche auch mit geschweiften Klammern begrenzt wird.

Im ersten Block werden die Berechnungen durchgeführt, die für den Status in Figur A aus Abbildung 17 benötigt werden. Ausgehend vom Codebeispiel in Listing 20 ist bekannt, dass der Oberschenkel durch die Arkustangensfunktion immer entlang der Geraden „b“ zum Kontrollobjekt ausgerichtet ist. Somit ist es möglich, in jeder Position des Kontrollobjektes den Winkel Alpha, in Abbildung 17 zu berechnen. Voraussetzung dafür ist die Länge jeder einzelnen Seite. Genau diese drei Geraden werden als Erstes in Listing 21 nach der If-Anweisung deklariert. Da alle Positionen dem gleichen Koordinatensystem zugrunde liegen, bestimmt man durch die Verwendung der Funktion „length“ die Länge der einzelnen Geraden. Wenn drei Seiten in einem beliebigen Dreieck bekannt sind, können nun die fehlenden Winkel durch den Kosinussatz berechnet werden. Die Grundform der Formel ist in Abbildung 19 an Erster Stelle abgebildet.



Figur A

Abbildung 18: IK-Schema

$$a^2 = b^2 + c^2 - 2bc \cdot \cos\alpha$$

$$\alpha = \operatorname{acos}\left(\frac{b^2 + c^2 - a^2}{2bc}\right)$$

Abbildung 19: Kosinussatz

Um nun den Winkel Alpha zu berechnen, muss die Formel nach Alpha umgestellt werden. In Abbildung 19 an zweiter Stelle steht die neue Formel, die durch das Lösen der Arkuskosinusfunktion das gewünschte Ergebnis liefert.

Bis auf einen kleinen Unterschied findet man genau diese Funktion auch in Listing 21 wieder. In diesem Fall ist die vordefinierte Funktion „clamp()“ noch vorgeschaltet. Um unmöglichen Positionen der Schenkel vorzubeugen wird der Lösungsbereich der Formel durch die Parameter „-1“ als Minimalwert und „1“ als Maximalwert beschränkt. Der neue Winkel wird in die Variable „AlphaX“ gespeichert.

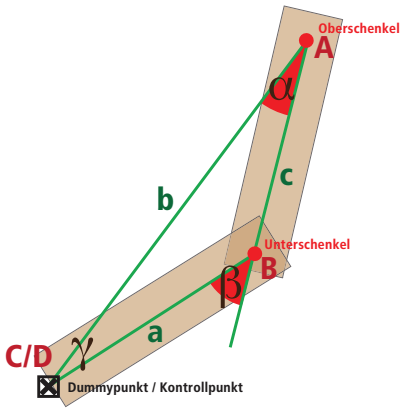
Jetzt wird die Richtung des Knickes ausgewertet, um gegebenenfalls den Winkel zu invertieren. Dies übernimmt wie auch schon vorher eine If-Anweisung. Wenn die Checkbox vom Nutzer angeklickt wurde, gibt diese den Wert eins zurück und bejaht somit die Aussage. In beiden Fällen wird die Winkelvariable „Alpha“ definiert. Diese setzt sich aus der Addition der errechneten Winkel „AlphaR“ und „AlphaX“ zusammen. Damit sich der Winkel im Wahrheitszweig umdreht, wird „AlphaX“ zusätzlich mit „-1“ multipliziert.

Da in Berechnungen mit Winkelfunktionen das Ergebnis immer in Radianten wiedergegeben wird, muss als letzter Schritt die Winkelumwandlungsfunktion auf die Variable „Alpha“ angewandt werden. Gleichzeitig werden nun auch die Körperrotation und die Hüftrotation vom Endwinkel abgezogen.

Tritt der Fall ein, dass der Vergleich der beiden Längenvariablen eine falsche Aussage liefert, springt die Expression in die nächste Else-Anweisung. Diese spiegelt Figur B aus Abbildung 17 wider. Da in diesem Status keine Winkel berechnet werden müssen, weil alle Gelenke gestreckt sind, hat nur die Arkuskosinusfunktion Einfluss auf die IK-Kette. Aufgrund der Tatsache, dass Variable „AlphaDeg“ schon in Winkelgrad umgerechnet wurde, müssen nur noch die Körperrotation und die Hüftrotation abgezogen werden.

```
if (laengeX < laenge)
     {
        a =          length(B,D);
        b =          length(A,C);
        c =          length(A,B);

        beta =       Math.acos( clamp( ( a * a + c * c - b * b ) / ( 2 * a * c ) , -1 , 1 ) );

        if(flip == 1){x = radiansToDegrees(beta * -1 + Math.PI);}
        else{x = radiansToDegrees(beta + Math.PI);}
     }
else
     {   x = 0;}
```

Listing 22: Beta-Winkel-Expression



Figur A

Abbildung 20: IK-Schema

Der erste Winkel der Inversen Kinematik Kette ist fertig. Nun folgt die Berechnung des Unterschenkelwinkels, wobei dieser eine eigene Expression für die Rotationseigenschaft erhält. Die Initialisierung der Variablen deckt sich mit der des Oberschenkels, somit ändert sich nur der Inhalt der If-Abfrage für den Längenvergleich.

Listing 22 beschränkt sich auf den Teil der sich gegenüber der Expression, die im oberen Element verwendet wird, abgrenzt. Da alle Komponenten im gleichen Koordinatensystem liegen und alle Seiten des Dreiecks (ABD) angegeben sind, muss nur die Arkuskosiunsfunktion nach „Beta“ umgestellt werden. Auch hier wird wieder zur Fehlervorbeugung die Begrenzungsfunktion eingesetzt. Danach folgt die Abfrage der Flipvariable. Durch eine wahre Aussage wird zum Winkel „Beta“ „-1“ multipliziert. Bis auf diesen Punkt sind die Aussagen der If-Anweisungen gleich. Es wird direkt die Umwandlungsfunktion integriert und zum errechneten Winkel „beta“ „PI“ hinzuaddiert. Das ist nötig, damit der Winkel nicht oben sondern wie In Abbildung 20 unten beginnt. Außerhalb der Umrechnungsklammer von Radianten zu Grad wäre es auch möglich gewesen 180° anstelle von „PI“ zu addieren.

Wenn die Länge zwischen Oberschenkelgelenk und Kontrollobjekt länger ist als die addierte Länge der beiden Schenkel, dann gibt die Expression für den Winkel eine Null zurück. Bis die Bedingung wieder erfüllt ist richtet sich der Unterschenkel aufgrund der Eltern-Kind-Beziehung immer entlang der Geraden „b“ aus.

Die beiden Expressions können auf alle IK-Ketten mit einem Gelenk angewandt werden. Es ist nur darauf zu achten, dass die anfangs eingeführten Variablen den neuen Ebenen nach aktualisiert werden.

7.9.4 Fersen- und Ballenexpression

Damit die fertige Figur wirklich sicher auftreten und abrollen kann, ohne dabei zu verrutschen sind viele Expressions, Effekte und Hilfsobjekte nötig. Im weiteren Verlauf wird der zu erarbeitende Mechanismus mit dessen Elementen erklärt.

7.9.4.1 Fersen- und Ballenmechanismus

Das Ziel der Methode ist es, die Kontrolle des Fußes so simpel wie möglich zu halten. Wie in Abbildung 21 eingezeichnet, gibt es nur einen Expressionregler um die Roll- und Auftrittsbewegung flüssig darzustellen. Figur A zeigt die Position, welche der Fuß einnimmt wenn am Regler positive Werte eingestellt werden. Der Fuß rotiert um den Ballenankerpunkt nach oben, ändert gleichzeitig jedoch auch die Position des Unterschenkels. Dies geschieht unter der Voraussetzung, dass die Inverse Kinematik eingehalten wird. Zeigt der Expressionregler, wie in Figur B, Null Grad an, so steht der Fuß flach auf dem Boden. Mit negativen Werten hingegen verhält es sich umgekehrt, denn hier wechselt der Rotationsursprung und der Fuß dreht sich um den Fersenankerpunkt, was in Figur C dargestellt ist. Auch hier wird der Rest des Beines wieder dynamisch mitbewegt.

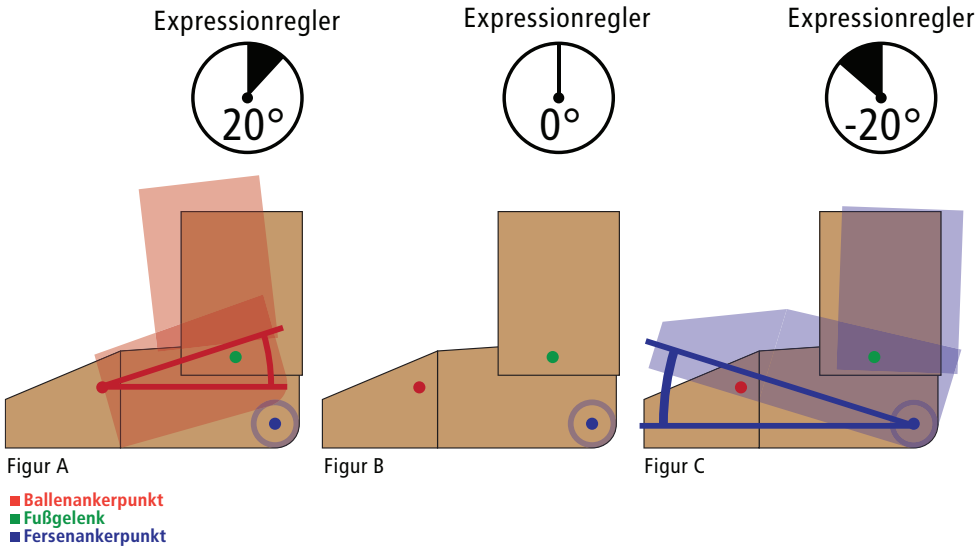


Abbildung 21: Abroll- und Auftrittmechanik

7.9.4.2 Transformationseffekte

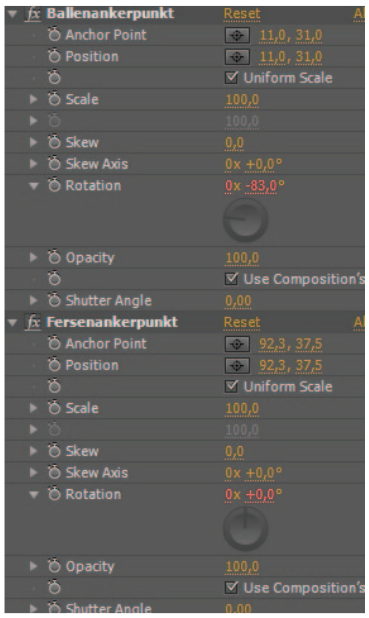


Abbildung 22: Transformationseffekte

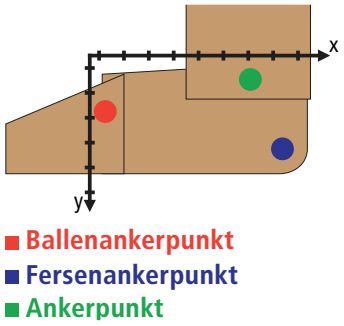


Abbildung 23: Ankerpunktkoordinaten

Um einer Ebene mehrere Transformationsursprünge zu verschaffen sind weitere Effekte nötig. Abbildung 22 zeigt die zwei essentiellen Effekte damit überhaupt ein Abrollen und Aufsetzen des Fußes funktioniert. Die Effekte „Ballenankerpunkt“ und „Fersenankerpunkt“ wurden der Fußebene hinzugefügt und sind Transformationseffekte. Dieser spezielle Effekt hat die Eigenschaft, Transformationsdaten wie Position und Rotation einer Ebene zu verändern, ohne dabei Einfluss auf die Originalwerte zu nehmen.

Als Merkmal ist zu beachten, dass die Punkte im Koordinatensystem des Fußes liegen. Wie in Abbildung 23 dargestellt können die drei wichtigen Punkte in diesem Koordinatensystem abgebildet werden. Damit der Fuß beim Hinzufügen der Effekte an der gleichen Stelle bleibt und nur der jeweilige Rotationsursprung verschoben wird, müssen die Werte aus dem Skript auf den Ankerpunkt und die Position gleichermaßen übertragen werden. Das ist auch in den Effektpaletten in Abbildung 22 zu sehen.

7.9.4.3 Dynamischer Ankerpunkt

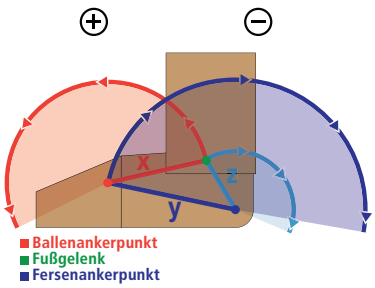


Abbildung 24: Schema für Winkelverläufe

Der Ankerpunkt, der in Abbildung 24 grün gekennzeichnet ist, existiert nicht als Rotationsursprung in der Fußebene. Dieser Punkt wurde in der Ankerpunktphase vom Nutzer als Fußgelenk festgelegt, welcher aber hier als Letztes Glied der IK-Kette fungiert. Außerdem ist er das Bindeglied zwischen Bein und Fuß. Der Fuß selbst kann und soll sich nicht um den Punkt drehen, da alle Positionen auch mit der Kombination aus Fersen- und Ballenrotation möglich sind. Dadurch erhält der Nutzer ein sehr stabiles Fuß-Rig, welches

besonders bei Laufanimationen seine Vorteile ausspielt.

Das Problem was sich nun darstellt ist, dass die Transformationseffekte keine Änderungen an Originalwerten zulassen. Dadurch ist es auch nicht möglich die Rotation an untergeordnete Elemente, was in diesem Fall der Zeh gewesen wäre, weiterzugeben.

Als Lösung müssen die globalen Positionswerte hergenommen werden um daraus Rotationspfade zu erstellen. Da keine globalen Rotationswerte existieren, wird im nächsten Schritt die Möglichkeit der Transformation von Positionswerten in Rotationswerte erläutert.

7.9.4.4 Einheitskreis

Als Ausgangswert steht der Expressionregler zur Verfügung, der auch die Ballen- und Fersenrotation steuert. Dieser Regler, welcher Winkelwerte ausgibt, kann genutzt werden um auf einer Kreisbahn, wie sie in Abbildung 25 schematisch dargestellt ist, Positionswerte abzubilden. Die hier angestrebte Methode nennt sich in der Mathematik Einheitskreis.[36]

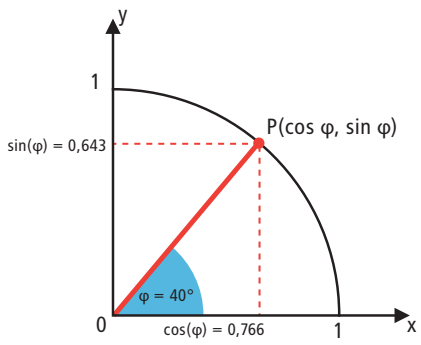


Abbildung 25: Einheitskreis

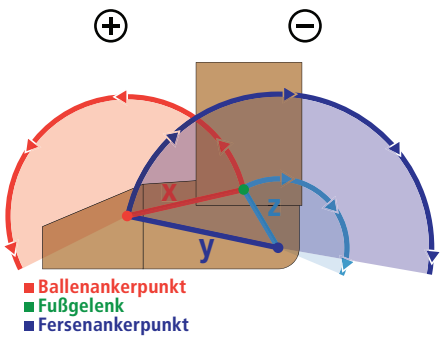


Abbildung 26: Schema der Winkelverläufe

Der Einheitskreis ist ein Kreis mit dem Radius 1 und einem Mittelpunkt der in einem kartesischen Koordinatensystem mit dem Punkt (0|0) übereinstimmt. Kann man von diesem Nullpunkt aus eine Gerade (r) zu einem Punkt (P) auf diesem Kreis bilden, bei dem ein Winkel (phi) durch die x-Achse und Gerade (r) begrenzt wird, so gilt für den Punkt P: yP = sin phi, xP = cos phi. Es ist also möglich unter Verwendung der Winkelfunktionen Sinus und Kosinus Winkelwerte auf Positionswerte abzubilden. Damit der Kreis einen beliebigen Radius einnehmen kann werden folgende zwei Formeln in den weiteren Expressions verwendet.

Der Radius der Kreise wird von der Entfernung der einzelnen Ankerpunkte zueinander bestimmt, was in Abbildung 26 dargestellt wird. Bei negativen Winkelwerten des Expressionreglers rotiert der Fuß um den Fersenankerpunkt. Dabei muss die Position des Zehs und die des Fußdummys auf ei-

36 Formeln und Tabellen, 35

ner Kreisbahn abgebildet werden. Wenn der Expressionregler positive Werte annimmt dreht sich der Fuß um den Ballenankerpunkt, wobei wieder die Position des Fußdummys anhand der Winkelfunktion neu gesetzt wird.

7.9.4.5 Vorbereitung vor Berechnung

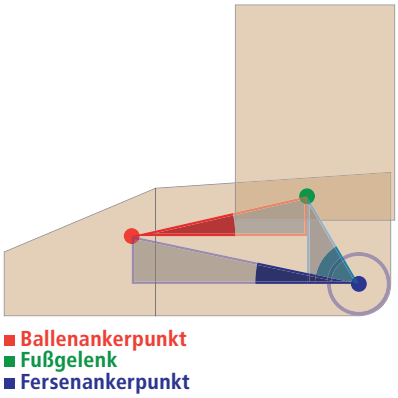


Abbildung 27: Winkelschema

Abbildung 28: Effektpalette mit Hilfsobjekten

Da die drei Punkte aufgrund der dynamischen Charaktererstellung bei jeder Figur anders aufgestellt sind, ändern sich auch die Ausgangswinkel zueinander, wie in Abbildung 27 erkennbar. Die Winkel sind immer zwischen der x-Achse und der Geraden zwischen zwei Punkten eingeschlossen. Wenn alle Winkel den Ausgangswert Null hätten würde sich das ganze Fuß-Rig verschieben. Deshalb müssen als Erster Schritt, bevor überhaupt eine Kreisexpression geschrieben werden kann, die in Abbildung 28 aufgezeigten Winkel berechnet werden, damit die dynamischen Differenzen ausgeglichen werden.

Abbildung 28 enthält die Effekte, die für die weitere Verwendung in einem Hilfsobjekt zwischengespeichert werden. Unter anderem sind auch die fehlenden Kreiswinkel und Positionen der Ankerpunkte enthalten. Neben den Positionswerten der drei Ankerpunkte, die direkt aus dem Skript in den Effekt übertragen werden, findet man auch die Ausgangswinkel, welche live aus den Ankerpunktpositionen berechnet werden.

7.9.4.6 Winkelexpression

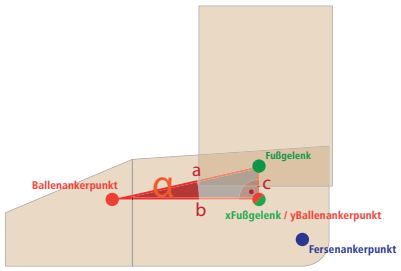


Abbildung 29: Ballenausgangswinkel

```
ball    = effect(,ball pt')(,Point');
ankle   = effect(,ankle pt')(,Point');

p1      = ankle;
p2      = ball;
p3      = [ankle[0],ball[1]];

a       = length(p1, p2);
b       = length(p2, p3);
x       = Math.acos(b/a);

if(effect(,ball pt')(,Point')[1] < effect(,ankle pt')
(,Point')[1])
        {
radiansToDegrees(x)*(-1);
        }
else
        {
radiansToDegrees(x);
        }
```

Listing 23: Ausgangswinkel (Ballen)

Zum besseren Verständnis ist in Abbildung 29 neben der Expression des „Ausgangswinkel des Ballenankerpunktes“ auch ein Schema der gegebenen Elemente abgebildet. In der Expression wird wieder mit der Deklaration der Variablen begonnen. Für das, in Figur A, gegebene Dreieck wird der Ballenankerpunkt und der Fußgelenkpunkt benötigt. Die Berechnung des gesuchten Winkels benötigt mindestens zwei Seiten des Dreiecks. Da von einem rechtwinkligen Dreieck ausgegangen wird, kann der dritte Punkt aus den zwei vorhandenen gebildet werden. Dies geschieht indem der x-Wert des Fußgelenks und der y-Wert des Ballenankerpunktes einen neuen Punkt generieren. Die drei Punkte werden als Nächstes in der Expression als „p1“ bis „p3“ gespeichert. Nun können die Längen der Seiten (p1,p2) und (p2,p3) durch die vorgegebene Funktion „length“ in die Variablen „a“ und „b“ geschrieben werden. Es sind jetzt alle Werte gegeben, um den gesuchten Winkel über eine trigonometrische Formel auszurechnen. Damit der Winkel ausgerechnet werden, kann muss die Formel wie nachstehend umgestellt werden. Das Ergebnis wird in der Variable „alpha“ gespeichert. Als Letztes folgt eine If-Abfrage, welche die beiden y-Werte der Ankerpunkte vergleicht. Die Aussage ist wahr wenn der y-Wert des Fußgelenkpunktes größer ist als der des Ballenankerpunktes. Wenn dieser Fall eintritt, wird beim Ausgeben des Winkels neben der Umwandlung in Grad auch eine „-1“ hinzumultipliziert, um den Winkel zu invertieren. In Abbildung 29 ist dieser Fall nicht eingetreten, da der y-Wert des Fußgelenks nicht den des Ballenankerpunktes überstiegen hat und dadurch die else-Anweisung ausgeführt wird. Diese unterscheidet sich nur dahingehend, dass man den Winkel nicht umgekehrt.

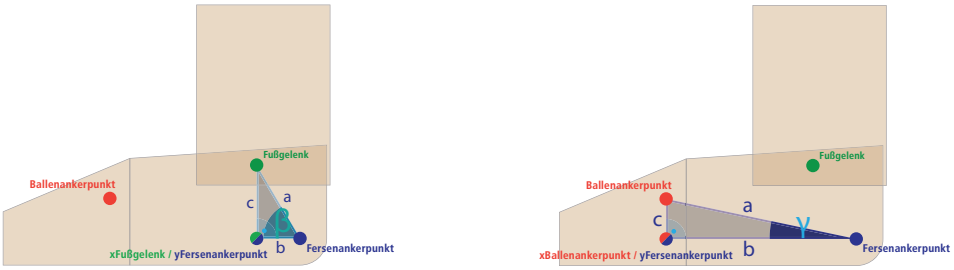


Abbildung 30: Schema für Ausgangswinkelberechnung

Für die anderen beiden Winkel wird, die in Listing 23 vorgestellte Expression analog verwendet. Wie Abbildung 30 zeigt, ändern sich nur die vorgegebenen Ankerpunkte mit denen man den gesuchten Winkel ausrechnet.

7.9.4.7 Kreisexpression

Da jetzt die gesuchten Ausgangswinkel feststehen, werden nun die Expressions für die eigentliche Rotation beschrieben. Der Fußanimator, als Objekt, das in der Hierarchie an erster Stelle steht, wird die Aufgabe haben, die Rotationswerte an die untergeordneten Fußnullobjekte weiterzugeben.

Insgesamt sind drei Rotationsexpressions nötig, die den Mechanismus steuern. Um eine Abrollbewegung zu erreichen muss der Fußanimator um den Ballenankerpunkt rotieren, wobei sich gleichzeitig auch der Fuß um die gegebene Gradzahl um den Ballen dreht. Die anderen zwei Expressions werden gebraucht um die Auftrittbewegung auszuführen. Dabei müssen Expressions für die Rotation des Fußanimators und des Zehs um den Fersenankerpunkt geschrieben werden. Aufgrund der Tatsache, dass bei der Rotation des Fußes durch einen Transformationseffekt keine untergeordneten Objekte, wie der Zeh, mitgeführt werden, erhält dieser für die Drehung um die Ferse eine zusätzliche Expression. In Abbildung 26 kann man sehen, dass insgesamt sechs Expressions nötig sind, jeweils drei für jeden Fuß. Um die Positionswerte auf die Ebenen ausgeben zu können, werden dieses Mal Punkteregler verwendet.

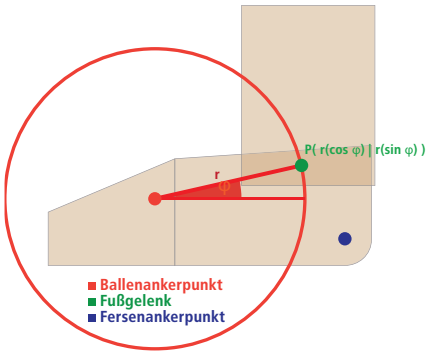


Abbildung 31: Ballenkreis

```
ankle      = effect(,ankle pt‘)(,Point‘);
ball       = effect(,ball pt‘)(,Point‘);
heel       = effect(,heel pt‘)(,Point‘);
w          = thisComp.layer(,control_master‘).
effect(,L_roll‘)(,Angle‘);
wx         = effect(,roll angle‘)(,Angle‘);

r          = length(ankle, ball);

wn         = w - wx;
wr         = degreesToRadians(wn);

x          = Math.cos(wr)*r;
y          = Math.sin(wr)*r;
diff       = ankle + ball - heel;

if(wn > - wx)
    {
    [0,0];
    }
else
    {
    [x,y] + diff;
    }
```

Listing 24: Kreisexpression

Die Abbildung 31 stellt den Abrollzustand und Listing 24 dessen Rotationsexpression dar. Wie gehabt initialisiert die Expression als Erstes alle Werte, die für die Positionsberechnung nötig sind. Den Anfang machen die drei bekannten Ankerpunkte, gefolgt vom Expressionregler, welcher die gesamte Prozedur steuern wird und durch die Variable „w“ dargestellt wird. Die letzte Variable „wx“, beinhaltet den zuvor berechneten Ausgangswinkel. Da man nicht im Einheitskreis mit dem Radius eins agiert, ermittelt die nächste Zeile die Länge zwischen Ferse und Ballen, was den Radius der gesuchten Kreisbahn bestimmt. Damit die Elemente die korrekte Ausgangsposition beibehalten, muss zunächst der Ausgangswinkel vom Expressionregler abgezogen werden. Der neue Winkel, der jetzt in Variable „wn“ gespeichert ist, wird nun für weitere Berechnungen durch die Umwandlungsfunktion in Radianten konvertiert. Dies ist nötig, weil im darauf folgenden Schritt die Positionswerte für x und y durch die umgestellte Einheitskreisformel berechnet werden. Für den x-Wert wird der Kosinus des umgewandelten Winkels berechnet und mit dem Radius multipliziert. Für die Berechnung des y-Wertes reicht es die Kosinus- mit der Sinusfunktion zu ersetzen.

Damit der Ursprung wie in Abbildung 31 dargestellt, im Ballenankerpunkt liegt, muss als Letztes eine Verschiebung des Rotationszentrums stattfinden. Dafür wird wieder eine neue Variable „diff“ eingeführt, welche sich aus der Addition des Fersenpunktes mit dem Ballenankerpunkt und der Subtraktion des Fußgelenkpunktes zusammensetzt. Zuletzt wird geklärt wann die Expression eingesetzt wird, da bei positiven Werten des Winkelreglers der Abrollwinkel aktiv ist und bei negativen Werten der Auftrittwinkel. Die If-Anweisung vergleicht den errechneten Winkel mit dem Ausgangswinkel, der im Expressionregler Null Grad darstellt. Ist der Ausgangswinkel kleiner als der berechnete, dann bleibt die Positionskorrektur des Fußanimators null. Tritt der andere Fall ein, so werden die errechneten x-und y-Werte aufgrund des Expressionreglers, der zwei Parameter als Ausgabe verlangt, in ein Array geschrieben. Damit die Kreisbahn im rich-

tigen Ursprung startet addiert die Expression zum Array die Differenzvariable.

Die zwei Auftrittexpressions werden analog dazu erstellt. Die Unterschiede belaufen sich lediglich auf die verwendeten Ausgangswinkel und die Verschiebung des Rotationsursprungs.

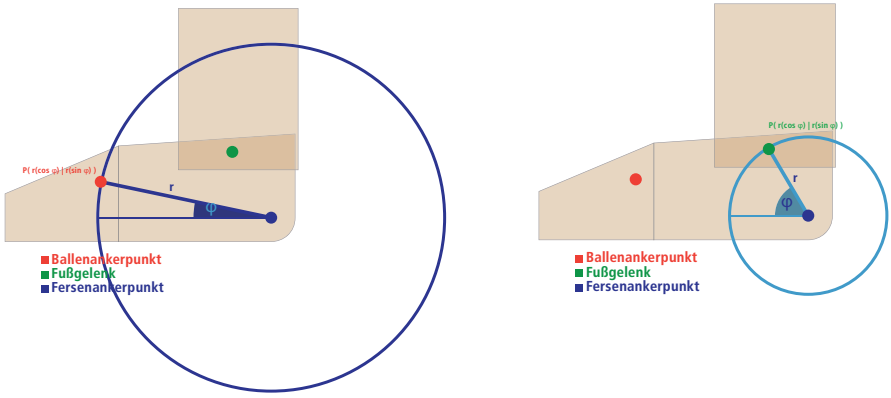


Abbildung 32: Fersenkreis und Fußgelenkkreis

7.9.4.8 Zehexpressions

Bisher wurden alle Berechnungen in einem Hilfsobjekt gesammelt. In diesem Abschnitt werden nun alle Werte der Hilfsebene in die dafür vorgesehenen Objekte eingearbeitet. Der Zeh, welcher als Erstes betrachtet wird erhält eine Expression für die Rotation und eine für die Position.

```
thisComp.layer(,Kontroll_Master').effect(,L_Zeh drehen')(,Winkel');
```

Listing 25: Rotationsexpression der Zehebene

Das Listing 25 stellt die Rotationsexpression dar. Hier wird der Zehrotationsregler der Kontrollebene mit dem Wert im Zeh verknüpft.

```
value + thisComp.layer(,helfer').effect(,L_Zeh_Anzeige')(,Punkt');
```

Listing 26: Positionsexpression der Zehebene

In Listing 26 erhält die Positionseigenschaft des Zehs die erste Kreisexpression aus der Hilfsebene. Dadurch ist der Zeh in der Lage sich um den Ballen zu drehen. Dabei wird die ursprüngliche Position nicht überschrieben, sondern mit Hilfe der Variable „value“ zum aktuellen Wert hinzuaddiert.

7.9.4.9 Fußexpressions

Als Nächstes erhält der Fuß insgesamt drei Expressions. Zwei davon sind für die jeweiligen Rotationen um den Ballen und die Ferse vorgesehen und eine um die Position zu steuern. Die Rotationsexpressions werden in die jeweilige Rotationseigenschaft der Transformationseffekte eingefügt. Dabei unterscheiden sich die beiden Expressions nur geringfügig. Als Ausgangspunkt für die Expression des Ballenankerpunktes dient der Abroll-und Auftrittsregler der Kontrollebene. Im Abschnitt zur Erklärung des Mechanismus wurde festgehalten, dass bei einem positiven Wert des Expressionreglers der Rotationsursprung im Ballenankerpunkt liegt und bei negativen Werten im Fersenankerpunkt.

```
w = thisComp.layer(,Kontroll_Master‘).effect(,L_Ballen‘)(,Winkel‘);
if(w >= 0)      {
        x=0;
                }
else            {
        x = w;
                }
```

Listing 27: Rotationsexpression der Fußebene

Im Listining 27 wird in der ersten Zeile der Wert des Expressionreglers in die Variable „w“ gespeichert, welche man im Anschluss in einem If-Vergleich auswertet. Ist „w“ größer als null, werden keine Werte des Expressionreglers weitergegeben. Im umgekehrten Fall ist „x“ gleich dem Expressionregler und der Fuß dreht sich.

Die Expression für den Fersenankerpunkt ist weitgehend identisch wie in Listing 27. Den einzigen Unterschied findet man in der If-Anweisung, denn hier wird das „w“ nur weitergegeben wenn es kleiner als Null ist, sonst gibt die Expression wiederum null zurück.

```
Fussgelenk =            thisComp.layer(,helfer‘).effect(,Fussgelenk Pt‘)
                        (,Punkt‘);
Beindummy =             thisComp.layer(,L_Bein_Platzhalter‘).effect(,reale
                        Position‘)(,Punkt‘)
Ballenrotation =        thisComp.layer(,helfer‘).effect(,L_Ballen_Anzeige‘)
                        (,Punkt‘)
Fersenrotation =        thisComp.layer(,helfer‘).effect(,L_Ferse_Anzeige‘)
                        (,Punkt‘);

x = Fussgelenk+Beindummy-Ballenrotation+Fersenrotation;
```

Listing 28: Positionsexpression der Fußebene

Nun fehlt noch die Expression für die Fußposition. Der Fuß soll einerseits dem Fußdummy untergeordnet sein und dessen Position übernehmen, damit er einfach durch den Fußanimator bewegt werden kann, aber andererseits still stehen, damit er bei der Abroll- und Auftrittbewegung nicht mit animiert wird. Im Listing 28 ist dies wie folgt gelöst. Zunächst werden wieder alle Variablen zusammengetragen. Diese wären im Einzelnen die Position des Fußgelenks innerhalb des Fußes, gefolgt von der realen Position des

Fußdummyobjektes. Die letzten beiden Variablen speichern die Positionswerte der Ballenrotation und Fersenrotation, welche in den Kreisexpressions berechnet wurden. Zunächst wird die Position des Fußdummys an den Fuß weitergegeben, was zur Folge hat, dass alle Änderungen durch den Fußanimator auch an den Fuß übergehen. Wenn nun die Kreisexpressions zusätzlich Einfluss auf den Fußanimator nehmen, um die IK-Kette beim Auftreten oder Abrollen mitzuführen, darf der Fuß keine Positionsänderung annehmen. Deshalb wird als Nächstes die Ballenrotationsexpression von der Fußdummyposition abgezogen. Daraus folgt, dass sich die Positionswerte beim Abrollen aufheben, sodass der Fuß an Ort und Stelle bleibt. Damit dies auch beim Auftreten funktioniert wird die Fersenrotationsexpression addiert. Da diese Werte nun ausgeglichen werden, nimmt der Fuß nur Positionsänderungen an, die vom Nutzer über den Fußanimator erfolgen. Damit nicht die Rotationen vom Nullpunkt ausgehen wird außerdem die Position des Fußgelenks addiert.

7.9.4.10 Fußanimator

Zur vollen Funktionstüchtigkeit des Rigs fehlt noch eine Expression die den Abroll- und Auftrittmechanismus mit der Inversen Kinematik in Einklang bringt. Diese Expression wird in die Eigenschaft Position des Fußanimators eingebunden.

```
ankle   = thisComp.layer(,helper‘).effect(,ankle pt‘)(,Point‘);
ball    = thisComp.layer(,helper‘).effect(,ball pt‘)(,Point‘);
heel    = thisComp.layer(,helper‘).effect(,heel pt‘)(,Point‘);
r       = length(ankle, ball);
wx      = thisComp.layer(,helper‘).effect(,roll angle‘)(,Angle‘);
wr      = degreesToRadians(wx);

BallRotation = thisComp.layer(,helper‘).effect(,L_Roll_Display‘)
          (,Point‘)
HeelRotation = thisComp.layer(,helper‘).effect(,L_Heel_Display‘)
                 (,Point‘)

x       = Math.cos(wr)*r;
y       = Math.sin(wr)*r;
diff    = ankle+ball-heel;

z       = [(x+diff[0]),(y-diff[1])*-1];
value - HeelRotation + BallRotation - z;
```

Listing 29: Positionsexpression des Fußanimators

Wie gehabt sieht man in Listing 29, dass auch hier als Erstes die Variablen initialisiert werden. Dazu gehören die Ankerpunkte sowie die Länge zwischen Fußgelenk und Ballenpunkt. Außerdem findet auch der Ballenausgangswinkel den Weg in die Expression, genauso wie die Kreisexpression des Fußballens und der Ferse.

Nun schließt sich der Teil an in dem die nötigen Berechnungen erfolgen. Den Anfang macht eine weitere Kreisformel, die als Ergebnis die Position aus dem Ballenausgangswinkel und dem Radius der Strecke zwischen Fußgelenk und Ballenankerpunkt errechnet. Danach wird die Variable „diff“ berechnet, die wieder für die richtige Position des Kreismittelpunktes verantwortlich ist. In der Variable „z“ erfolgt die Berechnung der neuen Koordinaten, auf der Grundlage der Verschiebung durch die „diff“-Variable. Als

Letztes werden nun alle Werte zusammengeführt. Damit der Nutzer den Fußanimator bewegen kann und dieser nicht durch die Expression blockiert wird, beginnt die Zeile mit der aktuellen Position in der Variable „value“. Danach wird die Fersenrotation abgezogen und die Ballenrotation addiert. Dies hat zur Folge, dass der Kontroller die Kreisbahnen abfährt und dabei auch alle Unterobjekte, wie die Inverse Kinematik mitführt. Da der Fußanimator nicht immer an der gleichen Stelle sitzt, muss nun noch die Variable „z“ abgezogen werden, um die richtige Ausgangsposition zu gewährleisten.

7.10 Aufräumen der Komposition

```
comp.layer(langStr(Ap_heel)).remove();

for (var i = 8; i <= comp.numLayers; i++)
        {
        comp.layer(i).shy = true;
        comp.layer(i).locked = true;
        }

comp.hideShyLayers = true;
```

Listing 30: Aufräumen der Komposition

Der Nutzer kann jetzt mit dem Rig seinen Charakter animieren, aber um das noch komfortabler zu gestalten, wird nun die Komposition automatisch vom Skript aufgeräumt. Da der Fersenankerpunktdummy seinen Zweck erfüllt hat, wird er als Erstes In Listing 30 entfernt. Danach werden alle Ebenen, die nicht zur Animation des Charakters benötigt werden, ausgeblendet und für den Nutzer gesperrt. Diesen Prozess erledigt eine Schleife, die alle Ebenen ab der Nummer acht durchläuft und gleichzeitig ausblendet und sperrt. Die letzte Zeile gibt den Befehl, dass alle Ebenen mit dem Attribut „shy“ ausgeblendet werden.

7.11 Ausführen der Benutzeroberfläche

```
if (pal instanceof Window)
        {
        pal.center();
        pal.show();
        }
        else
        {
        pal.layout.layout(true);
        }
        pal.gr.ct3.g14.abtBtn.onClick = function ()
        {
        Window.alert(langStr(A_about));
        };


        this.run = function (thisObj)
        {
        this.buildUI(thisObj);
        };
```

Listing 31: Benutzeroberfläche und About-Button generieren

Im letzten Teil des Skriptes werden nun die Befehle aufgerufen, die das Skript in After Effects darstellen. Dafür wird zunächst mit einer If-Abfrage geklärt ob die Variable „pal“, welche den Interfacestring speichert, eine zulässige Fensterstruktur beinhaltet. Ist diese Aussage wahr, dann wird das Skript mit dem Befehl „center()“ in die Mitte gerückt und mit dem Befehl „show()“ in After Effects dargestellt.

Danach wird die Funktion deklariert, die ausgeführt wird wenn der Nutzer auf den Aboutknopf drückt. Dabei soll sich ein Systemdialog öffnen, der den am Anfang bestimmten String anzeigt.

Durch den Befehl „run“, der das Skript startet wird auch die letzte Funktion im Skript ausgeführt, die mit dem Befehl „buildUI“ für den Aufbau der Benutzeroberfläche zuständig ist.

8 Auswertung

Im letzten Kapitel dieser Arbeit soll nun das programmierte Skript unter die Lupe genommen werden. Besonders wird dabei auf die Kriterien des Pflichtenhefts und der Möglichkeiten der Charakteranimation unter Berücksichtigung der Animationsprinzipien eingegangen. Unter anderem spielen auch im weiteren Verlauf die Auswertung eines Betatest und Optimierungs- und Erweiterungsmöglichkeiten eine Rolle.

8.1 Pflichtenheft

Musskriterien

Zu den Punkten die zufriedenstellend erfüllt wurden, gehören der dynamische Charakteraufbau und die Möglichkeit, Verbesserungen nach dem Riggen vorzunehmen. Für das Zusammensetzen der Figur wurden die benötigten konstanten Werte auf ein Minimum reduziert. Nur die Größe der Komposition und der einzelnen Körperteile reicht dem Skript aus, um den Charakter zusammenzusetzen. Aufgrund der relativen Abhängigkeiten der Elemente zueinander sind alle erdenklichen Körperproportionen realisierbar. Funktionstüchtig ist auch die Optimierung des Charakters nach dem Riggen. Es ist möglich, Arme und Beine auch nach der Prozedur in Ihrer Position am Körper zu ändern, wobei alle Expressions im Takt bleiben. Auch das Hinzufügen von zusätzlichen Elementen ist durch das Parentingsystem von After Effects kein Problem. Man muss jedoch bei Änderungen innerhalb der IK-Ketten aufpassen, denn hier sind abnormale Verhaltensweisen bei Korrekturen nicht ausgeschlossen. Die Stabilität als Ganzes ist zufriedenstellend, wobei bessere Möglichkeiten für Änderungen in der IK und des Abroll-und Auftrittmechanismus erarbeitet werden sollten. Wenn alle Ankerpunkte immer innerhalb der Platzierungszonen gesetzt wurden, kann davon ausgegangen werden, dass es keine Posen der Figur gibt, welche Expressions dazu veranlasst abzustürzen. Allerdings können die permanenten Berechnungen der vielen Expressions dazu führen, dass der Prozessor bei älteren Computern überdurchschnittlich belastet wird. Das kann wiederum Auswirkungen auf die Arbeitsgeschwindigkeit haben. Im Fokus stand allerdings die Benutzerfreundlichkeit, der auch am meisten Planung gewidmet wurde. Das Skript ist in eindeutige Schritte untergliedert, die für den Nutzer leicht verständlich sind. Dabei existieren in jeder Phase Hilfsmittel, die das Arbeiten erleichtern. Durch die einfache Bedienung konnte im gleichen Zuge auch die Geschwindigkeit extrem gesteigert werden. Ein geübter Nutzer benötigt im Schnitt, mit einer vorgegebenen Photoshopdatei, fünf Minuten bis der Charakter fertig ist zum Animieren.

Sollkriterien

Die Fersen- und Ballenfunktion konnte mit etwas größerem Aufwand umgesetzt werden. Dies war auch nötig, um sich von den bestehenden Lösungen noch weiter ab-

zuheben. Auch die Mehrsprachigkeit konnte mit wenigen Problemen eingebunden werden, so dass nun eine größere Zielgruppe von dem Skript profitieren kann. Bis auf Japanisch und Arabisch werden alle Sprachversionen unterstützt. Zur Erweiterbarkeit können keine konkreten Aussagen getroffen werden, dafür müsste der Code selbst optimiert und angepasst werden. Es ist möglich, zusätzliche Funktionen wie die Integration des Puppettools einzubinden, jedoch muss dabei ein relativ großer Aufwand betrieben werden.

Kannkriterien

Die Unterstützung des Puppettools und das Riggen von Vielbeinern konnte aufgrund der zeitlich intensiven Erarbeitung der Abroll-und Auftrittsfunktion nicht umgesetzt werden. Bei einer späteren Einbindung in das Skript würden diese Features dem Nutzer noch mehr Freiheit geben und ein noch größeres Publikum ansprechen.

8.2 Betatest

Der Betatest startete nachdem eine funktionstüchtige Version des Skriptes vorlag. Den Nutzern wurde so die Chance gegeben selbst Hand anzulegen und zu experimentieren. Dabei wurde nicht der original Sourcecode veröffentlicht, sondern eine binäre Version des Skriptes, die nur verschlüsselt geöffnet werden kann. Um den Nutzern ein paar Hilfestellungen zu geben, was mit dem Skript möglich ist, wurden einige Videos produziert, welche die grundlegenden Funktionen erläutern.

Nachdem sich einige Nutzer gemeldet hatten und Rückmeldung zu dem Skript gaben, sind einige Hürden besonders aufgefallen. Zunächst wurden diverse Bezeichnungen in den unterschiedlichen Sprachvarianten berichtigt, die in manchen Fällen Fehler hervorrufen konnten. Zusätzlich wurde ein gravierender Fehler berichtigt, der nur Körperebenen mit ganzzahligen Maßen zuließ. Die größte Resonanz gab es jedoch in Bezug auf die fehlenden Rückmeldungen des Programms. Vielen Nutzern war nicht klar, warum das Skript manchmal unvorhersehbare Dinge tat. Besonders während der Aufbauphase sollten noch mehr Fehler, durch Hinweise, für den Nutzer abgefangen werden. Hier ist es wichtig, dass zum Beispiel die Ebenen einer Inversen Kinematik Kette vor dem Verbinden in einer vertikalen Linie ausgerichtet sind.

8.3 Charakteranimationsprinzipien

Fast alle Prinzipien der Charakteranimation sind auch mit dem Skript umsetzbar. Die Außnahme ist dabei das „Squash und Stretch“-Prinzip, was aufgrund der Expressions nicht sauber funktioniert. Weitere Einschränkungen gibt es in der Ansicht des Charakters, welcher immer nur aus einer Perspektive betrachtet werden kann. Um nun Drehungen umzusetzen, ist das Riggen des selben Charakters, aber in einer anderen Ausrichtung nötig. Außerdem sind Posen, welche der z-Richtung entlang in den Raum

zeigen nicht möglich.

8.4 Optimierungs- und Erweiterungsmöglichkeiten

Der aktuelle Code ist im Moment in dem Status, dass er funktioniert. Als größtes Manko kann man die vielen Prorgammwiederholungen ansehen, welche durch den Einsatz von Schleifen und Funktionen verkürzt werden können, was wiederum sehr zur allgemeinen Lesbarkeit und Übersicht beitragen würde. Weiterer Optimierung bedarf das Abfangen von nicht gültigen Nutzereingaben. Besonders der Betatest hat auf diesen Missstand hingewiesen. Bis auf die Anleitungsvideos fehlen noch einsteigerfreundliche Maßnahmen. Für viele Nutzer wäre eine Frage-Antwort-Sektion oder ein Handbuch, um Funktionen des Skripts nachlesen zu können, eine große Hilfe.

Wenn die Optimierungen abgeschlossen sind, können Erweiterungen, wie die Integration des Puppettools, angegangen werden. Durch das Puppettool ist es möglich organischere Charaktere zu erstellen. Da das Tool im Grunde nur ein Deformer ist, der die Ebene anhand von gesetzten Punkten verzerrt, benötigt man für eine IK-Kette nur eine Ebene und drei Verzerrpunkte. Dabei stellt jeder dieser Punkte ein Gelenk der IK-Kette dar. Zusätzlich zum Puppettool wäre die Unterstützung beliebig vieler Beine und Arme ein großer Mehrwert für das Skript. Dadurch steigert sich die Freiheit des Nutzers erheblich und unterschiedlichere Charaktere könnten geriggt werden. Um auch die letzten Sprachversionen von After Effects zu unterstützen, müssten die Sprachstrings für Japanisch und Arabisch eingebunden werden.

8.5 Fazit

Für mich als Grafiker hat sich das Skript im Arbeitsalltag schon bewährt. Als Ausgangspunkt galt das Problem der aufwendigen Animation von Animatics, welche durch das Skript erleichtert werden sollte. Nicht nur in der Geschwindigkeit auch im Detailgrad der Animationen konnte das Niveau der Animatics gesteigert werden. Die einzige Vorbereitung, die getroffen werden muss, ist das Zurechtschneiden der Körperteile. Da diese aber meist schon vom Illustrator in Einzelteilen angeliefert werden, ist dies nur ein geringes Übel. Nicht nur für Animatics ist das Skript geeignet, sondern auch für einfache Charakteranimationen zur Ergänzung von Grafiken. Da würde sich eine professionelle 2D-Animationssoftware nicht lohnen, wobei allein der Preis für diesen Aufwand zu hoch und die Kompatibilität zum restlichen Projekt in After Effects zu gering wäre.

Der Betatest hat gezeigt, dass Nutzer, die sich mit dem Skript noch nie vorher auseinander gesetzt haben, zunächst etwas verwirrt sind. Dieses Problem ist bekannt und wird mit der höchsten Priorität zur Verbesserung behandelt. Dennoch bietet das Skript eine sehr gute Grundlage um Optimierungen und Erweiterungen fortzuführen. Im Vergleich zu den anderen Varianten der Charakteranimation, die es in After Effects gibt

(Punkt 5.4) sticht es dennoch mit den ganz eigenen Features heraus. Um das Skript noch weiter zu verbessern und weitere Anregungen zu erhalten wird das Skript den Nutzern unter dem Namen „RigIt“ kostenlos unter der Website (http://www.rigitscript.com) zum herunterladen zur Verfügung gestellt.

Im Anhang befindet sich neben einem Bildschirmfoto der Website auch eine CD-Rom, die Beispielanimationen und eine komplette Videoanleitung beinhaltet. Außerdem ist der gesamte Quellcode als Text- und Javadatei und eine Installationsanleitung vorhanden.

Quellenverzeichnis

Bücher

Adobe Systems: Adobe After Effects CS3 Professional SCRIPTING GUIDE. San Jose 2007

Adobe Systems: JAVASCRIPT TOOLS GUIDE. San Jose 2011

Allen, Eric und Murdock, Kelly L.: Body Language: Advanced 3D Character Rigging. Indianapolis 2008

Brinkmann, Ron: The Art and Science of Digital Compositing: Techniques for Visual Effects, Animation and Motion Graphics. Burlington 2008

Cabrera, Cheryl: An Essential Introduction to Maya Character Rigging. Oxford 2008

DeHaan, Jen: Flash MX 2004 - Das offizielle Trainingsbuch. München 2004

Eßer, Kerstin Berit: Bewegung im Zeichentrick. Eine vergleichende Analyse öffentlich-rechtlicher Zeichentrick-Koproduktionen für das deutsche Kinderfernsehen unter ästhetischer und historischer Aspekte. 1997

Fontaine, Philippe: Adobe After Effects CS3, Das Praxisbuch zum Lernen und Nachschlagen. Bonn 2008

Geduld, Marcus: After Effects Expressions. Oxford 2009

Heller, Sabine: Charakter-Animation in Film und Fernsehen. Berlin 2004

Johnston, Ollie und Thomas, Frank: The Illusion of Life: Disney Animation. New York 1981

Meyer, Trish und Meyer Chris: Creating Motion Graphics with After Effects. 4. Auflage. Oxford 2008

Parent, Rick: Computer Animation: Algorithms and Techniques. Burlington 2007

Roberts, Steve: Character Animation: 2D Skills for Better 3D. Oxford 2007

Internetseiten

Adobe Systems: After Effects CS5.5 / Auswahlhilfe : Versionsvergleich. http://www.adobe.com/de/products/aftereffects/buying-guide.html, 01.07.2011

Adobe Systems: Flash Professional CS5.5 / Funktionen, http://www.adobe.com/de/products/flash/features.html, 20.07.2011

Adobe Systems: Verzerren von Objekten mit den Puppenwerkzeugen. 2007 http://www.adobe.com/de/designcenter/aftereffects/articles/aftcs3it_ciblesson8.html, 02.07.2011

Anime Studio: Anime Studio pro 8. Key Features, 2011 http://anime.smithmicro.com/asp-features.html, 20.07.2011

Autodesk Softimage 2011: What Is Nonlinear Animation?. 2011, http://softimage.wiki.softimage.com/xsidocs/nla_intro.htm#Rdv10200, 20.07.2011

Dufresne, Nicolas: Duik Tools : DuDuF IK Tools pour After Effects, http://www.duduf.com/ressources/duik/index.html, 16.07.2011

Ebbert, Dan: AE Scripting. 2005, http://www.motionscript.com/ae-scripting/introduction.html, 15.07.2011

Ebbert, Dan: Dan Ebbert‘s AE expressions lab, Inverse Kinematics. http://www.motionscript.com/expressions-lab-ae65/ik.html, 15.07.2011

Frantz, Matt: Changing Over Time: The Future of Motion Graphics. 26.05.2003, http://www.mattfrantz.com/thesisandresearch/motiongraphics.html, 01.07.2011

Gartner, Kert: Detailierte Anleitung zum Knotenbasierten Compositing. 24.09.2010, http://mediablog.tmaekler.de/tag/node-based-compositing/, 17.07.2011

Hedin, Henric: Comparison of Node Based Versus Layer Based Compositing, 2010 http://hig.diva-portal.org/smash/record.jsf?pid=diva2:327273, 17.07.2011

SPJA, EX: The Digital Animation Primer, 2001, http://www.ex.org/4.2/09-column_bts1.html, 20.07.2011

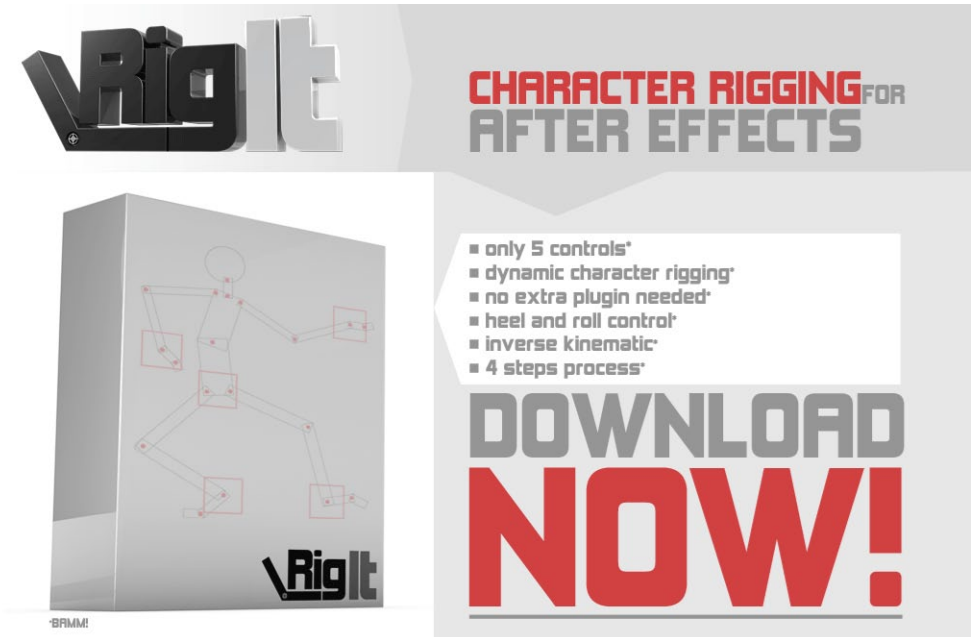
Toon Boom: Animate Pro. The most complete animation software for professionals. Key Features. 2011, http://beta.toonboom.com/professionals/animate-pro/features, 20.07.2011

Anlagen

CD-Rom Inhalt

- Skriptquellcode
- Installationsanleitung
- Schritt-für-Schritt-Anleitungsvideo
- Animationsbeispiele
- Puppettool-Animationsbeispiele
- Website (http://www.rigitscript.com)

Website



Anlage 1: Rigit-website (http://www.rigitscript.com)

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe. Alle Teile, die wörtlich oder sinngemäß einer Veröffentlichung entstammen, sind als solche kenntlich gemacht. Die Arbeit wurde noch nicht veröffentlicht oder einer anderen Prüfungsbehörde vorgelegt.

(Ort), (Datum)

(Unterschrift)